Proposal: Discovering Unaligned Expectations in Universities and Industry for New Graduates in Computer Science and Software Engineering

Alan Franzoni
Georgia Institute of Technology
Trieste, Italy
alan.franzoni@gatech.edu

Hasti Ghabel
Georgia Institute of Technology
Atlanta, GA
hghabel1@gatech.edu

INTRODUCTION

There is a widespread agreement that new graduates from computer science and software engineering **do not always possess required skills, abilities or knowledge when joining the tech industry**: a lot of entry-level jobs actually require three years of experience [3]; gaps between Engineering Education, and Practice (what an engineer does in real life) do exist [12]; the software industry presents dissatisfaction in relation to the level of recently graduated professionals [10]; there is considerable room for improvement in what is taught to software students [in relation with job relevance] [8]; many employers find that graduates and sandwich students come to them poorly prepared for the every day problems encountered at the workplace [4].

Some universities and programs even took steps to try and fix this problem in some specific classes by doing all kind of things: from purposely hindering and disrupting the software development processes [4], to adapting and incorporating industry training strategies into a software engineering course [10], to creating and adapting a project-based software engineering course that led the students to face with current, real-world engineering problems [6], and to highlight to students how relevant is having and developing critical soft skills to succeed in projects[2].

At first, we thought that different outcomes could come from different programs, so we explored the difference between Computer Science and Software Engineering programs, but those didn't prove really relevant; the official ACM/IEEE curricula for Computer Science [7] and Software Engineering [1], which many universities base their program on, are somewhat overlapping, and some studies trying to highlight differences in outcomes between CS and SE graduates were mostly inconclusive: a lot of core competencies are shared [9] [11]. And, those recently-updated curricula don't seem to incorporate lessons from the aforementioned efforts.

The acknowledgment of this skill gap and the efforts to train new graduates for the industry go back as far as 1992 [5]. So, **if in a quarter of a century little to nothing changed, what is the real matter**?

RESEARCH QUESTION

Our question is: **does the perceived skill gap in fresh graduates exist because the academy is unable to provide a good training, or just because a) the academy is not even trying to do that kind of job, and b) the industry is taking that kind of job for granted, or c) the students think they should be getting something that the university has no intention to provide them with?**

THE HYPOTHESIS

We think that:

- Most students, when picking their major, have little to no idea what they're going to actually study, and they probably expect to learn mostly about programming and creating applications;
- Most teachers, when designing their courses, think about teaching what they deem useful to achieve the so-called *computational thinking* in their students;
- Most professionals and employers, when hiring fresh graduates, expect they'll be able to immediately and fully carry out whatever real-world task is assigned to them.

So, our hypothesis is that one of the reasons for the perceived skill gap is that all those that should - in an employer's view - care for learning some skills to be used at work, don't actually have that aim during their education phase. We think that the expectations among three groups differ. These three groups are composed of **students, educators and school staff, and industry professional**. The main problem is not only that one side hasn't enough resources or skills to achieve a certain goal, but, rather, that there's a different vision or gaol on what should be done, and different and unaligned *rewards* exist for different groups.

CHI'16, May 07–12, 2016, San Jose, CA, USA

ISBN 123-4567-24-567/08/06…$15.00

DOI: http://dx.doi.org/10.475/123_4

RESEARCH METHOD

It would be great to have one-on-one interviews with a lot of people, and ask them what were their expectations before starting their degree and after getting it, to understand professionals' point of view, and to try to understand what teachers and university staff designing courses is actually trying to do; but, that would involve an important amount of time, both for the researchers and all the people involved.

Still, we think we can leverage modern technology to achieve something useful; we plan to **create a survey** where we ask questions to assess the thoughts of students, university teachers, and industry professionals to discover what it is, in their opinion, the current goal of university degrees (noticeable Bachelor's degrees in Computer Science, Software Engineering or whatever a similar degree is called in one's country), and how that affects job proficiency and chances of being hired. We employ a mix of open and multiple choices questions.

Independent variables:

- Category: CS/SE student, CS/SE professor/teacher/university staff, industry professional in the tech/software field
- Country of residence
- Country where somebody got his/her degree (if any)
- Previous programming skills or actual job experience in the tech/software field before starting university;
- If employed, company size and tech department size;

Dependent variables:

All the variables are about a CS/SE bachelor's degree fresh graduates.

- Self-reported current goal for bachelor's degree: what they think that university aims at, right now;
- Self-reported ideal goal for bachelor's degree: what they wish that university would do, if different from current;
- Self-reported expected GPA relevance for actual job proficiency;
- Self-reported university ranking relevance - how the ranking of a well-known university can impact the job qualification?
- Self-reported expected (or actual, for professionals) time to achieve full job proficiency when entering the industry (stop being in a junior role)?
- Self-reported expected time to land the first job for a current fresh graduate?
- Self-reported top skill which they think useful at a job that can't be taught at school (if any)
- Self-reported expected proficiency of graduates versus practitioners with a relevant work experience in the same ballpark (4 years) but without a degree, when getting a job;
- Self-reported expected proficiency of graduates versus practitioners with a relevant work experience in the same ballpark (4 years) but without a degree, after 5 years;

The latest two question try to discover whether the perceived gap could actually be "reversed" - so, whether the graduates think they're up for a slow start when compared with practitioners, but are then able to gain the same (or more) speed.

The survey will actually be slightly tweaked depending on the target, mostly for wording's sake.

What we perceive is novel is that **we target all categories at the same time**, so we can put their answers in perspective, and that we will **aim for a larger group than previous studies**; we'd really like to reach 300+ answers to our survey.

RECRUITMENT

Since we would like to get a good sample, we'll create a website, where we describe what we are doing, and we'll provide links to the survey and an easy way to share the website itself; we'll setup a mailing list where everybody can sign up and get follow ups on our research progress and results. We will use SurveyMonkey or Google Forms as the tool to put our survey questions in it and collect the results. We hope that a bit of advertising/social media push will help us getting to the numbers we need.

We think this research is relevant, since many of our peers giving us feedback thought it was very interesting, so we'll ask our classmates to share the website as much as they can, and we'll ask each survey-taker to share it as well, especially to categories that are different than himself (e.g. "Share this survey with your professors!")

DATA ANALYSIS

Once we collect the data, we'll graph and analyze it, and we'll try to uncover patterns that highlight the relations between some our variables. Of course, that wouldn't necessarily imply a causal relation. We'll be using an ANOVA test to verify the statistical significance of our results.

LIMITATIONS

Threats to validity, both internal and external: we hope to get a good sample from around the world, but there's always the risk we incur in some biases - e.g. if some universities or organizations get really involved with the research we may get too many results from those specific institutions. We may also get a skewed number of answers from a specific target group - we'll need to understand what a good proportion is between groups as well, because we doubt the world offers the same amount of professionals, teachers, and students.

Additionally, there will be a limitation on finding a large number of participants that meet our group categories. It is more challenging to find a large number of teachers, professors, and also recruiters, since they have a busy schedule or they may not be allowed to participate in certain surveys due to their company policies.

Another thing is time limitation. We only have 4-5 weeks to provide the survey questions, send it to the people (small and

large group), validate the questions, and analyze and summerize the results. Sometimes, it takes **a couple of weeks** for each participant to only complete the survey and collect the results.

Also, we may be getting skewed answers with respect to institutions; for example, we may get a lot of answers from students from a certain university, some teachers from a different one, and then we may get a lot of practitioners that had no contact with such universities, and hence have no such perception; it would be great to match students and teacher from certain institutions, and then let see what employers and professionals say about such students; but this would require quite more time and a far more complex setup than what we can afford for this research.

DELIVERABLES

We collect the deliverables and their due dates in the table below.

Deliverable	Due Dates (year 2018)
Five weekly status reports	Every week, 06/25 - 07/23
Intermediate milestone 1	07/2
Intermediate milestone 2	07/16
Final presentation	07/30
Final project	07/30

Table 1. Schedule of each deliverable

TASKS

Below, we listed the tasks that needs to be completed for this project. We displayed the number of hours for each task in front of it inside the parenthesis.

Milestone 1

Task 0: Collaborate with teammates and complete the **project proposal**. (20 h)

Task 1: Create a **website** that explains our project objectives to the readers and links to our different surveys. This website is required to enlarge the audience with different experiences in the field of computer science and software engineering. (15 h)

Task 2: Plan the **survey questions** that target different categories including students, teachers and professors, industry professional and employers. The survey questions are very important to lead the research to its goal. Correspondingly, the given answers from different categories can provide a good and promising results analyses. (20 h)

Task 3: Using **proper tools** to provide the survey. Some available tools that we consider are Google Docs, or Monkey surveys. (10 h)

Task 4: Make the **survey** available to a smaller sample of categories. This gives us the opportunity to better understand what we missed in our survey or what was extra, which does not give us important information. We are planning to share the survey at this level with our classmates at Educational Technology - Summer 2018. We will use Piazza platform to share our survey with this group of people. (15 h)

Milestone 2

Task 5: The next step is to **validate** and **update** the survey questions based on the result and feedback from small group survey. (20 h)

Task 6: Create a plan on how to send our results to the people who are interested to know what we find in our research. We can setup a **mailchimp mailing list**. (15 h)

Task 7: Share the **validated survey with a large group of people**. We need to share it on different forums, blogs, linkedIn, facebook, OMSCS slack channel, or maybe targeting employers and professionals in some companies. (15 h)

Task 8: **Collect the results** from our survey and describe the results. We will do research on available tools that we can analyze our results and display them on a graph to effectively describe our findings. (20 h)

Final Steps

Task 9: Put all our findings together and **prepare the final project paper**. (30 h)

Task 10: Complete the **final project** and **final project presentation**. (20 h)

WEEKLY MILESTONES

Here, we describe the weekly due dates, goals, project-related assignments, and associated tasks due dates.

Week #5

- June 11, 2018 - June 18, 2018:
  - **Goal**: Complete the project proposal.
  - **Assignments Due**: Project Proposal
  - **Tasks**: Task 0

Week #6

- June 18, 2018 - June 25, 2018:
  - **Goal**: Provide the website to store the surveys, Provide survey questions.
  - **Assignments Due**: Weekly Status Check 1
  - **Tasks**: Task 1 & Task 2

Week #7

- June 25, 2018 - July 2, 2018:
  - **Goal**: Create the online survey on best possible platform.
  - **Assignments Due**: Weekly Status Check 2, Intermediate Milestone 1
  - **Tasks**: Task 3 & Task 4

Week #8

- July 2, 2018 - July 9, 2018:
  - **Goal**: Validate survey and find solutions on automatically share results with people.

- **Assignments Due**: Weekly Status Check 3
- **Tasks**: Task 5 & Task 6

Week #9

- July 9, 2018 - July 16, 2018:
  - **Goal**: Send the online survey for the larger group of people and Collect data.
  - **Assignments Due**: Weekly Status Check 4, Intermediate Milestone 2
  - **Tasks**: Task 7 & Task 8

Week #10

- July 16, 2018 - July 23, 2018:
  - **Goal**: Continue on collecting results, summerize and analyze the collected results to prepare them for the final paper.
  - **Assignments Due**: Weekly Status Check 5
  - **Tasks**: Task 8 & Task 9

Week #11

- July 23, 2018 - July 30, 2018:
  - **Goal**: Complete the Final Project and Final Presentation
  - **Assignments Due**: Final Project, Final Paper, Final Presentation
  - **Tasks**: Task 9 & Task 10

MEMBERS OF THE TEAM

The name and GT username of the team members are mentioned below:

1. Alan Franzoni : *afranzoni3*
2. Hasti Ghabel : *hghabelzadeh3*

TASK DIVISION

The tasks are divided among the team members. The number of each task is described in front of the name of each team member.

1. Alan Franzoni Task #s: 0, 1, 2, 5, 6, 7, 8, 9, 10
2. Hasti Ghabel Task #s: 0, 2, 3, 4, 5, 7, 8, 9, 10

REFERENCES

1. Mark Ardis, David Budgen, Gregory W. Hislop, Jeff Offutt, Mark Sebern, and Willem Visser. 2015. SE 2014: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering. *Computer* 48, 11 (nov 2015), 106–109. DOI: http://dx.doi.org/10.1109/mc.2015.345
2. Maria Cecilia Bastarrica, Daniel Perovich, and Maira Marques Samary. 2017. What Can Students Get from a Software Engineering Capstone Course?. In *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET)*. IEEE. DOI: http://dx.doi.org/10.1109/icse-seet.2017.15
3. K. Chakrabarti. 2018. 61 percent of Entry-Level Jobs Require 3 Years of Experience. (2018). https://goo.gl/u6wKSc
4. Ray Dawson. 2000. Twenty dirty tricks to train software engineers. In *Proceedings of the 22nd international conference on Software engineering - ICSE '00*. ACM Press. DOI:http://dx.doi.org/10.1145/337180.337204
5. R.J. Dawson, R.W. Newsham, and R.S. Kerridge. 1992. Introducing new software engineering graduates to the 'real world' at the GPT company. *Software Engineering Journal* 7, 3 (1992), 171. DOI: http://dx.doi.org/10.1049/sej.1992.0018
6. David Delgado, Alejandro Velasco, Jairo Aponte, and Andrian Marcus. 2017. Evolving a Project-Based Software Engineering Course: A Case Study. In *2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE&T)*. IEEE. DOI: http://dx.doi.org/10.1109/cseet.2017.22
7. ACM Computing Curricula Task Force (Ed.). 2013. *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. ACM, Inc. DOI: http://dx.doi.org/10.1145/2534860
8. Lethbridge. A survey of the relevance of computer science and software engineering education. In *Proceedings 11th Conference on Software Engineering Education*. IEEE Comput. Soc. DOI: http://dx.doi.org/10.1109/csee.1998.658300
9. F. Meziane and S. Vadera. 2004. A comparison of computer science and software engineering programmes in English universities. In *17th Conference on Software Engineering Education and Training, 2004. Proceedings.* IEEE. DOI: http://dx.doi.org/10.1109/csee.2004.1276512
10. Carlos Portela, Alexandre Vasconcelos, Sandro Oliveira, and Mauricio Souza. 2017. The Use of Industry Training Strategies in a Software Engineering Course: An Experience Report. In *2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE&T)*. IEEE. DOI:http://dx.doi.org/10.1109/cseet.2017.16
11. Ghulam Rasool and Touseef Tahir. 2014. A Comparison of Software Engineering and Computer Science Undergraduate Programs in Pakistan. In *2014 12th International Conference on Frontiers of Information Technology*. IEEE. DOI: http://dx.doi.org/10.1109/fit.2014.27
12. Vijayakumar Sivanesan. Analyzing the Gaps between Engineering Education and Practice. (????). https://gatech.instructure.com/files/638493/download?download_frd=1 EdTech Spring 2017.