Self-Portrayals of GI Junior Fellows

Dominik Schultes*

Teaching at a university of applied sciences

Abstract: A university of applied sciences focuses on both application and science. Before becoming a professor at a university of applied sciences, I got in touch with both worlds: the purely scientific one, doing research in the field of algorithm engineering, and the 'real' one, working in large software development projects. After making these experiences, I am convinced that the approach of a university of applied sciences is a very good one.

Keywords: Applied sciences, tertiary education.

ACM CCS: Social and professional topics → Professional topics → Computing education

DOI 10.1515/itit-2014-1059
Received May 31, 2014; accepted October 14, 2014

1 Introduction

I have always liked teaching. I started to give private lessons at secondary school, continued as a tutor when studying at university and later as a research associate. One of the nice things about teaching is the fact that you usually get a direct feedback on your work, either immediately for example as an intelligent question during the lesson that indicates that a certain point has been understood or with some delay for example in an exam where even a tricky task is solved correctly. Of course, there are cases of negative feedback as well, but the point is: there *is* feedback.

Since my appointment as a professor at the Technische Hochschule Mittelhessen – a university of applied sciences – teaching has been my main occupation. One of the biggest challenges in teaching is keeping the balance between practice and theory, between application and science. A university of applied sciences has integrated this challenge within its title in form of a compromise: applied sciences. By this, it is situated in the middle of the available educational institutions, where universities form the one extreme and industrial training the other one. My own professional experience in three very different positions confirms that this compromise is a very good approach. In the next sections I will elaborate on this.

2 Research

As a research associate I developed route planning algorithms that are able to preprocess road networks of a whole continent and, then, compute the provably shortest path from a given start to a given target point within a few microseconds on a standard server machine [1–3]. This is more than one million times faster than using the classic algorithm by Dijkstra [4]. To be one million times faster than the classic solution was a good sales argument: several companies were very interested in my work and several cooperations were arranged. However, in each case I found it surprisingly difficult and tedious to come to an agreement. Not some deficiency of the algorithm was the obstacle – nor the money. In fact, there were a lot of small, from a scientist point of view quite unimportant obstacles, for example existing data structures – which were suboptimal anyway – where the algorithm should fit in or an existing project plan that had no slot to try a new algorithm. Once I got the reply "No, we cannot try to integrate the algorithm now, a while ago our project management has been changed to Scrum [5, 6] and during the next weeks, i. e., the current sprint, we are not allowed to assign new tasks to the team." I have to admit that at that point I was not aware of Scrum and I remember quite well my somewhat arrogant thoughts: "Oh my God, they are so unflexible. I've just presented such a great algorithm and they are not able to give it a try due to some fancy project management stuff." I concluded that after having studied computer science at three different universities and having worked as a research associate for some years on a quite practical topic, obviously there was still an important piece missing to establish understanding for the real-world problem of dealing with software development projects.

***Corresponding author: Dominik Schultes,** Technische Hochschule Mittelhessen, Friedberg, e-mail: dominik.schultes@iem.thm.de

3 Software development

Since I do not like 'missing pieces', I decided to leave academia and to start as a senior software engineer at Capgemini (formerly known as sd&m AG). Coincidentally, both my first and my last project at Capgemini had the ZDF – Zweites Deutsches Fernsehen – as client. The first project was the further development of the ZDFmediathek[1], the last one was a complete relaunch of the three main web portals, zdf.de, heute.de, and zdfsport.de. In the last project I acted as project leader. During that time I learned a lot of things that I had not learned in academia. In retrospect many things now seem to be self-evident, but for me they were not at that point. Let me give a few examples.

- There are a lot of unexpected events that can force you to change the project plan. My favorite example is the consideration of one colleague's vacation plans: his wife was pregnant and the beginning of the planned vacation was set to the expected day of birth. Unfortunately, it turned out that the baby was not willing to observe my carefully crafted project plan and decided to postpone the birth for two weeks which meant that my colleague wanted to postpone his vacation for two weeks which meant hat I had to make up a new work package.
- In academia, I worked on a few, quite difficult problems. At Capgemini, I usually was confronted with quite simple problems – at least if you look at them in an isolated way. However, there was a surprisingly big amount of such 'simple' problems, which made it quite hard to keep track of them and to link them in the right way.
- You have to deal with people whose words have different meanings than you expect them to. Here, my favorite example is the word 'day'. I had always thought that a day started at midnight. For the people at ZDF it goes without saying that a day starts at 05:30 in the morning. At first glance this seems to be complete nonsense. At second glance it is completely logical because the 'broadcasting day' starts at 05:30 and the program planning aligns to this point in time.[2] However, this leads to some confusing effects: when it is afternoon and a program entry should be displayed for a movie that starts in the night at 02:00, it must say "today at 02:00" – although "tomorrow" would be the correct term, at least in the non-TV-world. Interestingly, the ZDF anticipated that there would be several users that would complain about the somewhat incorrect wording and, therefore, defined a rule that for each program entry between 00:00 and 05:30 a little moon symbol should be displayed to indicate that the special night time logic applies. In the final phase of the project, one tester realized that the moon icon was not there. It was one of the most amusing conversations with our customer, the ZDF, where we discussed the question who to blame for losing the moon.
- There are always new ideas from the customer – or even more dangerously – from a highly motivated software developer in your own team. In order to get to *any* result, it is most important to guard the development team from spontaneous change requests – even if it is just a simple moon icon which should be added. At that point I realized that a few years ago it had been very sensible that our industrial partner had not integrated my algorithm immediately but sticked to the Scrum rules.

On the whole, I have realized that the challenges, priorities and processes within the industry are quite different from my previous experience in academia.

4 Applied sciences

After having been in touch with both science and application, I noticed that I liked both. Therefore, I became a professor at a university of applied sciences. I mainly teach in the media computer science program of study on subjects like web programming and mobile applications. The focus of the university of *applied* sciences is reflected by the facts that

- throughout the curriculum, beginning in the very first semester, you find several courses that apply the approach of project-based learning,
- frequently there are guest lectures by external experts,
- there is a mandatory industrial internship, and
- most bachelor and master theses are completed within a company.

Of course, this will not result in the same level of experience that you attain in a full-time job in a company, but, at least, it can be quite a good starting point. Nevertheless, particularly when teaching on topics like mobile applications I realize how fast the technology changes. It took

1 http://www.zdf.de/ZDFmediathek

2 Indeed, this turns out to be a very convenient feature: you do not have to turn the page of your TV guide at midnight, but you can keep sitting in a relaxed fashion for another 5.5 hours.

Android only three years to evolve from a market share of 3.9% to a share of 66.2% in 2012.[3] In the same time, the mobile operating system Symbian, the former leader w. r. t. market share, practically disappeared. It is also the same time it takes from teaching the course till the students might complete their studies. This observation confirms that it would be wrong to concentrate only on applications and concrete technologies, which might be outdated very soon. Thus, I conclude that the things I learned and dealt with as a research associate are still useful as well. For example, to be able to tell apart a less efficient algorithm or piece of program from a more efficient variant is useful basic knowledge, which does not expire very fast.

5 Conclusion

Surely, the concept of a university of applied sciences has not been invented by me. In fact, I was not even born in the year 1968 when a general notion of universities of applied sciences was agreed on in the Federal Republic of Germany. Nevertheless, I find it interesting that my own CV – the experience I gained at three different positions – exactly reassures that it is a good idea to have a type of university that focuses on applied sciences because both parts – application and science – need each other and, if you leave one of them out, there would be a missing piece.

References

1. H. Bast, S. Funke, P. Sanders, and D. Schultes. Fast routing in road networks with transit nodes. *Science*, 316(5824):566, 2007.
2. D. Schultes. Route planning in road networks. PhD Thesis, Universität Karlsruhe (TH), 2008.
3. D. Delling, P. Sanders, D. Schultes, and D. Wagner. Engineering route planning algorithms. In *Algorithmics of Large and Complex Networks*, volume 5515 of *LNCS*, pages 117–139. Springer, 2009.
4. E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.
5. H. Takeuchi and I. Nonaka. New New Product Development Game. *Harvard Business Review* 86116:137–146, 1986.
6. K. Schwaber and M. Beedle. Agile software development with Scrum. Prentice Hall, 2002.

3 w. r. t. worldwide mobile device sales to end users, cp. Gartner: http://www.gartner.com/newsroom/id/1543014, …/id/2017015, …/id/2120015, …/id/2237315, and …/id/2335616, retrieved in April 2014

Bionotes

Prof. Dr. Dominik Schultes
Fachbereich IEM, Technische Hochschule Mittelhessen, Wilhelm-Leuschner-Str. 13, 61169 Friedberg, Germany
dominik.schultes@iem.thm.de

Dominik Schultes completed his PhD thesis at the Karlsruhe Institute of Technology on “route planning in road networks”. His work was awarded with several prizes, in particular the Scientific American 50 Award and the Klaus Tschira Preis für verständliche Wissenschaft. He worked for Capgemini for more than three years and took part in several large web development projects for the ZDF (Zweites Deutsches Fernsehen). Since 2012 he is a professor at the Technische Hochschule Mittelhessen, focusing on web and mobile applications and algorithm engineering. Dominik serves as a junior fellow of the *Gesellschaft für Informatik* since September 2013.