The Nugues Approach to Project Courses

Görel Hedin
Department of Computer Science
Lund University
gorel.hedin@cs.lth.se

Abstract

Project courses at the Master of Science level is an excellent opportunity for students to do a small research-style project as a preparation for the Master thesis. Such projects can even lead to publications, as shown multiple times by Pierre Nugues in his supervision of student projects.

In this paper I describe the Nugues approach to student projects and how I use it for compiler projects. I also discuss IPSERC, an article structure I find useful for computer science students, to complement the IMRaD structure they usually learn in scientific writing classes.

1 Introduction

The field of computer science evolves rapidly, and students taking advanced courses are often eager to try out their new knowledge, constructing some new experimental tool, investigating a new technique, or solving some real-world problem. Doing this in the context of a research-oriented project course gives students an excellent preparation for their thesis project.

In this paper, I will describe the *Nugues approach*, i.e., Pierre Nugues's approach to running such projects. Pierre developed this approach for his projects in language technology, and I will also discuss how I use it for compiler projects, and how it came about that I started running such projects at all.

Key ideas in the Nugues approach include getting students started on both implementation and writing very quickly, and with weekly feedback on progression. This is important not the least as the department project course is short, running during a seven week study period. Another important part of the approach is that students write up a short research-style paper, and present it in a conference-like setting. Pierre even makes it an explicit goal for his students to potentially submit the paper to a real conference. I find this very inspiring, in particular since he has published several papers with his students based these projects, see, e.g., Grundström and Nugues (2014), Weegar et al. (2014), and Norrby and Nugues (2015).

A difficulty in supervising computer science students is that they often do not know what or how to write. Their training in scientific writing is typically focused on the IMRaD structure for experiments in medicine and natural science, with Introduction, Methods, Results, and Discussion. In the computer science field much of the research is focused on creating solutions to problems, and the IMRaD structure does not fit. This mismatch is something the Nugues approach helps counter: the students get to see typical computer science papers with sections on solutions and evaluation. I use the term *IPSERC* for this typical structure (Introduction, Problem, Solution, Evaluation, Related Work, Conclusion).

The reason that I and many others at our department came to run these student projects at all, was a bit peculiar. It was triggered by a bureaucratic decision that was taken by our faculty in 2013, namely that the department must give *fewer* courses than before. I describe this briefly in Section 2. Then I present the Nugues approach in Section 3, some adaptations I did for my compiler projects in Section 4, and the IPSERC structure and its relation to IMRaD in Section 5. Finally, I end with some concluding remarks in Section 6.

2 A bureaucratic decision

Working at the university is a constant struggle between trying to do good teaching and research, and trying to navigate the administration. In 2012, our faculty got a decrease in teaching funding, and at the same time there was an "overproduction" of

From Electrophoresis to Wikidata: Festschrift in honor of Pierre Nugues, pages 7–10.
Published by University of Tartu Library, May 2025.

student credits. I.e., the faculty was teaching more than it got paid for by the university (and in the end, by the government). The main reason for this overproduction was that the university had extended the engineering education from 4.5 years to 5 years, in order to adapt to the new Bologna model for education, and without being compensated for this by the government. The faculty decided, as one part in trying to solve the financial problems, to decrease the number of courses offered at the faculty, requiring all departments to reduce their number of ”course codes”, i.e., the number of courses offered. This did of course not change the number of credits that the students wanted to or needed to take, so I will leave any explanation of the logic behind this decision to future work.

This decision was very much at odds with our plans at the computer science department. The whole area was rapidly expanding worldwide, and we had too few advanced courses to offer an increasing number of interested students. Yet, even our department had to decrease the number of ”course codes”. How should we solve this?

At the time, we had three project courses: one in operating systems (EDAF01), one in intelligent systems (EDAN50), and one in language technology (EDAN60). The latter one was Pierre Nugues's course. We then got the brilliant idea to merge these courses into one new ”course code” (EDAN70). We could in practice continue teaching as before, but with a bit more administration as students would now sign up just for the project course, and we would need to spend some time on sorting them into topics.

Merging the project courses had an upside as well. Other instructors wanting to run student projects could easily join without having to set up their own course. I was one of them. I was teaching the compilers course, and wanted to start a course where students could do a compiler project. In the current situation, adding a new course was out of the question, but I could perhaps achieve something similar through the new merged project course.

3 The Nugues Approach

Since Pierre had been so successful in running his student projects, I interviewed him about his approach in the fall of 2014, before starting my first batch of projects. Here is what I noted down:

Prepared proposals. The instructor prepares project proposals beforehand, each with a short description, and presents them at an introductory meeting. Students can in general not propose their own project, but if someone has an idea, the instructor will consider turning it into a proper proposal.

Concrete product. Each project has a concrete goal: a product, on which some property can be measured.

Small groups. Students work either individually or in pairs.

Weekly deadlines. The instructor meets each student group every week and gives them concrete goals for the next week. Students demo results at each meeting.

Start reading and writing immediately. Already the first week, the students should write the first outline of their report, as well as read a related article.

Start implementing immediately. The students need to start the concrete implementation immediately, to get something running as soon as possible, no matter how small.

Baseline. Many projects are about improving efficiency or computing something with higher precision. The instructor helps the students find a suitable baseline for measurements.

Presentation before report. The course runs for 7 weeks, with presentations in week 7. The final report is then not due until a few weeks later, after the Christmas break.

Clear requirements. To pass the course, the students need to a) complete a 4-page double column research-style report (3 pages if they work individually), b) present the work in a 15 minute talk using 10 slides, and c) meet the weekly deadlines, producing what has been agreed on.

4 Adapting to compiler projects

The Nugues guidelines fitted very well with the compiler projects I had in mind, and I have run the compiler projects in this way since the start in 2014. One minor difference is that many of the compiler projects are about building new tools,

Project title
Short overall description
Illustration
3-4 Concrete steps/subgoals
Optional stretch goals
Evaluation suggestions
References to key articles
Name of supervisor

Table 1: Content of project proposal slide

Week 1	Introduction meeting, select project
Week 2	Report with draft intro and outline
Week 3	First demo
Week 4	First release of code
Week 5	Report extended with solution
Week 6	Draft of presentation slides
Week 7	Presentation at seminar
+4 weeks	Final report and code

Table 2: Weekly deadlines

sometimes interactive, and where there is no clear baseline to evaluate against. The important aspects to include in the evaluation can then be, for example, demonstration of that the approach works on interesting examples, and that the performance is sufficient for practical use.

In my research group, the PhD students had nice ideas for compiler projects, and to make things work in a streamlined way, I developed a template for a project proposal slide (Table 1) and a standardized set of weekly deadlines (Table 2). Each proposal includes a number of concrete steps to get going, and sometimes also stretch goals for the ambitious student. In addition to writing a project report, the compiler students also publish their code as open source.

5 IPSERC versus IMRaD

In medicin and natural science, the main goal of research is to find new knowledge about the existing world, typically following research methods to establish this knowledge. This fits well with the IMRaD structure of research articles. In engineering sciences, including computer science, the goal of research is often instead to invent new methods to perform a useful task. We might call such an invented method a *solution method* to distinguish it from a *research method*: A solution method performs some useful task whereas a research method helps establishing knowledge about some existing phenomenon. Of course, once a new solution method has been invented, it becomes part of the world. It is then interesting to analyze its properties, in which case the usual research methods can be useful.

Courses in scientific writing will usually point out the importance of looking at existing literature in the research field, to understand how to structure articles for that field. However, concrete advice is then typically focused on the IMRaD model only. This confuses engineering and computer science students, since IMRaD does not match the structure of the papers they read or what they are expected to write. For example, what is a ”method” in their research? Is it perhaps the way they developed their solution? Is it how they came up with the idea for their new solution? Or what is it?

To balance the omnipresent IMRaD structure, I therefore propose the IPSERC structure as a concrete article structure which many computer science and engineering papers follow. An IPSERC paper has an Introduction section that includes a Problem description. The meat of the paper is then a description of a suggested Solution to this problem, described in one or more sections. After this there is typically a section called Evaluation which describes various properties of the suggested solution. Often, the evaluation involves experiments, and IMRaD can then be useful for structuring the Evaluation section. After the Evaluation section there is often a Related Work section. While the Introduction will include a brief discussion of previous related work, the Related Work section can discuss similarities and differences in more detail, since it is placed after the Solution and Evaluation sections. Figure 1 summarizes the IPSERC structure and its relation to IMRaD.

I have found it very helpful to present this explicit IPSERC structure to computer science students. It makes it easier for them to recognize the structure of the papers they read, identifying both IPSERC and IMRaD elements in them, and making students more aware that different research areas have different traditions.

6 Concluding remarks

Seven weeks is a very short time to do both implementation, evaluation, and research-style report writing, but with the Nugues approach, it works

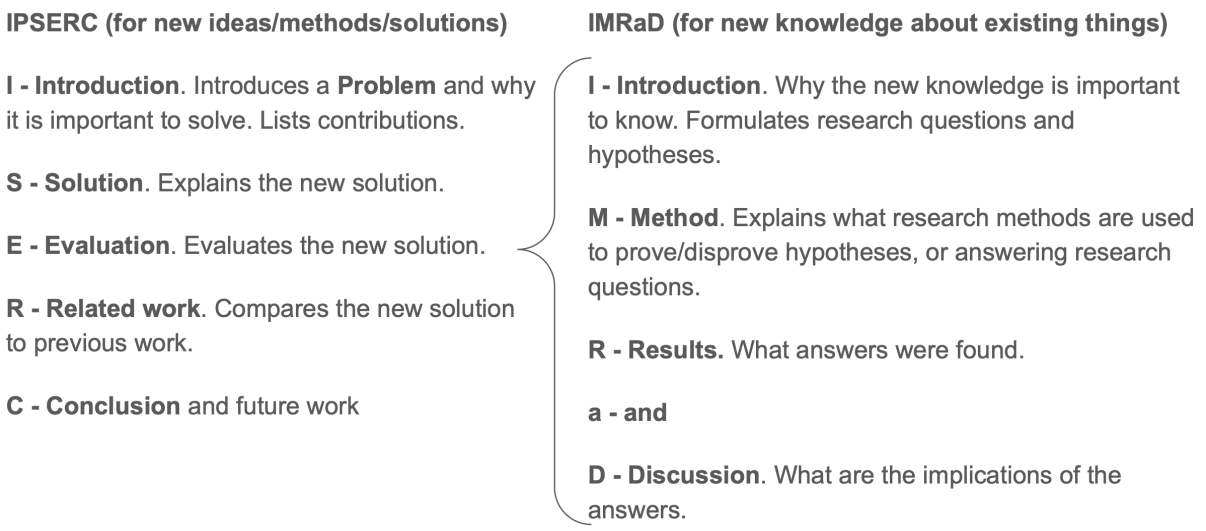

Figure 1: Typical IPSERC and IMRaD article structures. The Evaluation of an IPSERC article might be structured according to IMRaD.

very well. These projects are also very fruitful for PhD students, who get training in supervision before they take on supervising MSc projects. By the way, the inscrutable decision by the faculty to stop new courses in 2013 was lifted a few years later.

Acknowledgments

Thank you, Pierre Nugues, for sharing your ideas on this topic with me, and for all other interesting discussions we have had. Thanks to Boris Magnusson for feedback on an earlier draft of this paper.

References

Jakob Grundström and Pierre Nugues. 2014. Using syntactic features in answer reranking. In *AAAI 2014 Workshop on Cognitive Computing for Augmented Human Intelligence*, pages 13–19. AAAI.

Magnus Norrby and Pierre Nugues. 2015. Extraction of lethal events from wikipedia and a semantic repository. In *workshop on Semantic resources and semantic annotation for Natural Language Processing and the Digital Humanities at NODALIDA 2015*, pages 28–35. Linköping University Electronic Press.

Rebecka Weegar, Linus Hammarlund, Agnes Tegen, Magnus Oskarsson, Kalle Åström, and Pierre Nugues. 2014. Visual entity linking: A preliminary study. In *AAAI-14 Workshop on Computing for Augmented Human Intelligence*.