

Programming

COURSE EVALUATIONS FROM

CO-OPERATIVE EDUCATION PROJECT 2016 S1

## What is good about these courses

Alongside PR282 I did PR294 Server Side Programming where I was introduced to the client server model which helped for designing the SSM application (project). PR283 taught me C# and during it I spent more time trying to figure the ‘right’ or ‘nice’ way to build applications as opposed to trying to figure out how to get them to work at all. It also taught me some language features such as LINQ which the SSM application uses for finer control over iterations. PR301 continued to improve my architecture designs through use of patterns and common pitfalls or ‘bad smells’ to avoid in code.

PR203 gave me enough SQL skills that I could complete this project, albeit relying on T-SQL documentation heavily.

As far as Client-server model goes, connecting things to things, how could a developed program fit into the wider view. Only students who took PR294 got a taste of that kind of work, which I personally really liked, having my PHP application pull from a small but comparatively large database and serve to a web page really showed how things can work together.

Obviously, being a software development project, I made significant use of my knowledge from all previous software engineering courses, from SE101 to PR301, although interestingly not as much as might be expected. Since this project was focussed on web development, it didn’t utilise so much of the content which was largely focused more on object oriented programming. Regardless, these courses still provided the fundamentals and the more advanced nuances of programming and following best practices that were necessary for the success of this project. Principles like DRY (Don’t repeat yourself) and SoC (Separation of concerns) all came into play for this project.

The design and structure of the software for this project was similar to the Model-View-Controller (MVC) design architecture, first introduced to me in PR282 and further used in PR283 and PR301. Having an understanding of MVC helped in drawing parallels with the architecture we were using, which provided an easier and deeper understanding of how to use the architecture and the benefits that using it provided. I was also able to relate the architecture to my knowledge of Design Patterns gained from PR301. The architecture we used was in some ways similar to the “Façade” design pattern in that it provided a higher level abstraction (the API) of another sub-system (the DAO) to make it easier to use.

## Recommendations

I think the biggest drawback to the programming courses in general, is that at the end of the course the student only has 1 application, which I think is appropriate as a final result piece, but the learning that goes into it varies incredibly and I feel I would have learned more by having multiple, smaller applications each week (or two) as practice, in addition to the ‘main’ app the class develops.

The drawback to the one application approach is that the student can and will stumble through it, and by the end learn enough to say ‘I know better ways of doing this now’ but then lacks the opportunity to fix it. Smaller bite sized applications would help syntax/language stick to memory better and could provide a variety of differing requirements. During PR282, PR283 and PR294 we had syntax exercises, which could be interpreted as ‘small applications’ but what I had in mind was bigger.

Perhaps the ultimate goal would be requiring the student develop an assistant application that is needed to complete the final assignment for the class. If the final assignment was some kind of data processor, a helper app may be needed to retrieve the data and/or put the data into some kind of intermediary format that the final app is to use.

The client/server model is a much larger topic than is covered in PR294 or more generically I feel there is room to expand upon inter program connectivity, such as databases, web server or not covered: communication between processes.

One final thing I feel would be a great improvement is the creation of courses focusing on the hardware side of programming. With micro-controllers and internet of things aspects. The idea being that the wider view or bigger picture of how things can fit together translates into the physical world as well.

Synchronicity in programming is not a concept that was really covered in any of the previous courses, or if it was it was mentioned only briefly, and yet it seems like a rather important concept. Especially since it can be difficult to grasp at first, I think it would be beneficial to include AJAX or some other form of asynchronous programming as a part of one of the software engineering courses, perhaps PR301 since it’s a slightly more advanced concept.

I would also suggest possibly covering the use of libraries slightly more. While it was touched on in the courses, there weren’t really any assignments or projects that made use (or allowed use) of libraries in a way that actually provided experience in using them. I think it would be useful to provide that experience, especially in using libraries commonly used in the industry, such as jQuery. I can understand the need for students to create most things themselves for the sake of learning, especially since if it was allowed a lot of projects would be far too easy using libraries, but I believe encouraging or maybe even requiring the use a library in at least one assignment would be a good thing for providing some experience that can benefit students when they move into the work place.

My main block to achieving learning in this environment has been the ability to critically analyse what I’m learning. In earnest I’ve found that I learn best by given example, and practically having the chance to do that. As such, I would suggest providing group or team-work as a basis for learning. Like getting a BA, programmers, interface designers together for a project as a basis for achieving learning outcomes. I’m a big believer in team work as I’ve learnt so much from the project as it applies to teams.

Documentation is really important. If I hadn’t decided on documentation as a legitimate extension of my project, then I think we could have just got away with occasional testing. But because documentation was the centre of the quality management, then I realised that our testing wasn’t up to scratch. This lead to developing proper documentation for each prototype. My observation is that I didn’t know that distinction until now. Especially when making things by yourself, documentation gets over looked because you’re working with no one. Make documentation important in the classes. I would be willing to come and talk about it.

I suggest having classes that are reward driven and allowing people to pass or fail based on their competency, motivation based rewards that get applied during classes. I don’t have any concrete answers, except that I used to get motivated by making databases that test my knowledge – tap into my learning potential and thirst for more.