In the discussion of agile methods, I agree with the idea “While for some ‘agile methods’ are a step backwards...” Since I don’t have any experience in agile software development, the idea of agile methods makes me doubtful. In my opinion, non-computer related positions cannot understand software, architecture and coding. The requirements clients raised may seem easy to acquire during the communication, but eventually impossible to realize in software design. If the clients and the designers, or developers obtain opposite ideas and they can’t persuade each other, it will take many efforts to conduct compromise and communication. Furthermore, the agile methods make changes and produce new versions rapidly based on clients’ requirements. I think it is a huge waste of resources and increases the cost of a project development and deployment, as well as a heavy burden for designers to communicate with clients and meet the changing requirements.

I agree with the opinion in research directions that “choice and evaluation is the matter”. From my coding experience, each time I meet a problem, the first thing is go to the stackoverflow and look for a solution. Normally each post has a set of answers and I will pick the first working solution that solves my problem. However, I am not sure if that one is the best one(performance measure, logic, size, or etc) Therefore, a better method for decision making and evaluation would be a useful tool in the software design and architecture field. For example, artificial intelligence is a keen research area these years, imaging using AI to estimate the performance of all answers simultaneously. Or furthermore, use AI to design software and its architecture, so making decisions and evaluating won't be a problem for humans, designers and developers’ job is to provide ideas.

The words “ A ‘we will just fix it in the code’ attitude is far too prevalent” is very inspiring. It makes me think of my own experience. I encountered a bug a long time ago, spent more than 3 hours searching for solutions and thinking how to fix that bug. Although it finally worked, I didn’t make any notes but had a vague idea about the solution. After a few months, in a similar project I met the same problem again. I had an impression on that but I just couldn’t remember the solution. Eventually it took me another few hours to do the same thing. Since then, I took notes on my notebook each time I figured something out. Similarly, I think this idea is exactly the attitude in software design and architecture. There are so many failed projects in the world, only studying the weakness, problems and mistakes from them can make them useful. Concluding design rules, studying software history and fixing mistakes from old projects should be the right attitude and goal for software designer and architect.