# SE575

# V1: Software Architecture as Systems Dissolve, by Eoin Woods

Yiyun Zhang

10/5/2020

* I like the speaker’s explanation about software architecture’s evolution history, I think the design principles introduced in the class are the reflections of the software architecture’s evolution: they present the needs of the software architectures in old eras. Just like the words in the lecture notes: “A design is like wine, it takes a long time to see if it is good or not”, based on time and actual practice, these principles also represent designers’ expectations for the future in different phases. The monolithic is proven hard for people to maintain and scale as long as the scale of the monolithic is huge, therefore the designer concluded the design principle: ”Divide and conquer”. As for distributed monoliths, although they have different business functions, they still deploy as a whole. So I think the design principle “Increase cohesion and reduce coupling” is from this design fact. As the time goes by, mobile devices become popular while the internet connects all hardware together, many computer softwares needs to be moved to mobile platforms. Under this background, the designers realise design principles such as “Increase reusability, design for portability, etc” to meet the needs. I believe other design principles introduced in the class notes are also from the induction and conclusion from the previous architecture designs. It is a meaningful thing to review the history of software architecture’s evolution and think about how those design principles form.
* One thing I think the presenter can improve is by adding some real life examples, so those cases can help audiences better understand the logic and facts behind the talk. For example, when the presenter talks about the five distinct phases of software architecture’s evolution, he could list some examples such as: a .war file is an example of the monolithic architecture because it is a single file that contains all fundamentals of an application. If you want to buy the service, you have to buy everything contained in that file. For distributed monoliths, Apache ZooKeeper is a good example to explain the structure of distributed monoliths. It is a decentralized software that allows multiple clients access to it. For internet connected, smart phones such as android and ios are working examples that show vendors are selling non-functional requirement hardwares instead of applications themselves. For the internet as the system, he could give some SaaS examples such as Amazon Web services, Microsoft Azure, etc. These are services everyone is familiar with nowadays, audiences can pick up the idea easily by mentioning them. In conclusion, by adding these real examples, those people who experienced the five phases can understand the evolution of the software architecture in a more familiar way since they may have bought or heard those softwares and hardwares. While those who didn’t experience the five phases have some keywords to do research and figure out what exactly software vendors were selling in old times, in order to better understand the speaker’s introduction about software architecture’s evolution.
* In the past few years, artificial intelligence became a keen area of research. Many capitals and resources involved into the topic and artificial intelligence integrated with many technologies that already existed. So I think it is an unavoidable thing that artificial intelligence will be connected to the software architecture, just like the speaker’s prediction. From what I know, many huge technology companies such as Google and Oracle already use scripts to produce codes - developers no longer need to write code, only ideas. So in my understanding, in the future, with intelligence’s help, the computer can decide what is the architecture and how to construct it spontaneously: the developers’ job is to tell the computer what is the idea and goal, not coding or designing a specific form of software architecture. I am excited about this future and look forward to learning more about it.
* Enjoyed it, and I think it complemented the course. This talk not only helped me understand the history of software architecture’s evolution, but also predicted the future direction of development and which technology will be integrated. This video is very interesting to watch and think about.