Software Engineering for Industry

Topic 3: Software Architecture

Assessed Work - Groups of 3

Software architecture must be one of the most talked about yet most widely misunderstood topics in modern software engineering. On the one hand everyone talks about doing architecture and many people use the word “architect” in their job titles, yet very few people can clearly explain what software architecture is, what its goals are or what it entails in terms of activities and outcomes.

This week we are going to research some of the prominent views on software architecture and understand what is common and what differs between them in order to explore what the value of software architecture is, the different approaches to it, the core activities that characterise software architecture, and what outcomes we expect from software architecture work.

Joining us for the session will be Nick Rozanski (https://www.rozanski.org.uk) and Murat Erder (https://www.linkedin.com/in/murat-erder-62bb98). Nick is a very experienced software architect who today acts as software architect for systemically important international payments infrastructure at Mastercard. Murat is equally experienced and has spent a long period at Deutsche Bank, working on the architecture of critical internal services and applications for this 80,000 person bank. I have co-authored well-known software architecture books with both of them (https://www.viewpoints-and-perspectives.info and https://www.continuousarchitecture.info).

For this piece of work you should expect to spend about **6 hours in total**. This will comprise about two hours performing individual research and then an hour in your group pooling knowledge, comparing opinions and insights and consolidating what you know as a group. The exercise should then take about two hours of group work, performing some architecture work, with about another hour working together to answer some questions and produce some outputs.

Background Reading and Research (3 hours)

The goal of your research is to gain an understanding of what software architecture is and to decide if you think it is important, and if so why. My suggested approach to achieve this is to read some different opinions about and approaches to software architecture, gain an understanding of each, and then compare and contrast them in your groups to find commonalities and differences.

Martin Fowler, the Chief Scientist at ThoughtWorks and a prominent software development writer has a very popular page on his site devoted to software architecture (https://martinfowler.com/architecture/). As he says, when many people talk about software

architecture it often means “a hazily defined notion of the most important aspects of the internal design of a software system” and it would be fair to say that his page suffers from this a little too. However it is a useful starting point for a brief discussion of software architecture and many people point to it as a canonical source, given Fowler’s prominence in the industry.

To explore **what software architecture is** and its **essential activities**, in a little more depth, read chapters 2 and 7 of *Software Systems Architecture* (Rozanski and Woods, Addison Wesley, 2011) and chapters 1 and 2 of of the later book *Continuous Architecture in Practice* (Erder, Pureur and Woods, Addison Wesley, 2021). These four chapters are in an O’Reilly Learning playlist here. You can also find videos of Eoin Woods giving an introductory talk on Continuous Architecture on YouTube (e.g. https://www.youtube.com/watch?v=XEKwOuBesOg) and linked via www.eoinwoods.info.

Having explored the activities of software architecture, we need to remember that the term “software architecture” can refer to both the **activity** of doing architectural design and to the **artefacts** produced by those activities, so we now need to explore the artefacts of software architecture, which include models, styles, patterns and decisions.

One of the problems with creating software architecture models is that there are many dimensions to “the” architecture for a system, including its runtime functional components, its design time code modules, its deployment environment, the structure, ownership and distribution of data and so on. It is easy to produce models with these different dimensions tangled together, which are difficult to understand. The idea of “**viewpoints and views**” was invented to address this problem (first introduced in a short IEEE Software article by Philippe Kruchten, “*The 4+ 1 view model of architecture*”, available from Google Scholar).

A simple and widely used viewpoint set is the C4 Model. It was created about 10 years ago by Simon Brown, who is a well known software architecture consultant. He has developed a very nice page for people who want to start to learn about the approach at https://c4model.com. You can find examples of C4 in use on the C4 site and a few interesting ones via an internet search.

You may have noticed that a topic that the SSA book, CAiP book and Simon Brown’s writing all cover is the **quality attributes** of a system (also known as the “non functional requirements” - performance, scalability, security and similar aspects of a system). Most people in the software architecture field would agree that a key goal of software architecture is to meet the system’s quality attribute requirements. What did the material tell you about integrating quality attributes into your architecture work?

Finally, **architectural principles and decisions** have become more important in recent years as systems have become more distributed and more dynamic (I explain why in a bit more detail in the video linked above). Chapter 8 of the SSA book and chapter 2 of CAiP both discuss architectural principles and decisions and there is a well known Github site (https://adr.github.io) which describes a simple technique called “Architectural Decision Records”, for capturing architectural decisions, created (or perhaps rediscovered) by Michael Nygard in 2011: https://cognitect.com/blog/2011/11/15/documenting-architecture-decisions.html.

Start the research exercise by gathering as a group and deciding how to allocate the work. There is too much to do for everyone to do everything. Then spend a couple of hours on individual research before reconvening in your groups to share knowledge, compare insights and consolidate what you know.

Then move onto the exercise below.

Exercise

The Royal Albert Hall wants to develop a new booking system for tickets and we are going to help them with some software architecture assistance.

The hall hosts many different types of events, concerts etc and has a capacity of around 5000 seats. Customers can buy tickets online, by phone, or in person at the box office. For online or telephone booking, tickets are printed and delivered to the customer's home address.

Conscious of their brand, the Royal Albert Hall have all the normal concerns about security and privacy when selling online. They are also concerned about availability of the system and want to make sure that the website is always up and responsive. However at the same time they are naturally concerned about costs of developing and running the new system.

Generally, there are a few hundred bookings per day. But, on some days, a lot more people try to book. For example, when tickets are released for an event like the famous BBC Proms concerts, the site becomes overloaded and customers end up waiting in long virtual queues or simply have their requests rejected when they try to connect to the site. One of the main objectives for the redesign is to deal with these problems and to provide a smoother booking experience for customers.

An outline functional scope of the system includes:

- Displaying the programme of events and letting the user browse and search;
- Checking availability of tickets for a performance;
- Booking tickets for a performance, which involves choosing seats (or having the system choose them for you), taking payment via credit card, and providing download of e-tickets or dispatch of paper tickets.

The task for your group is to spend a couple of hours performing some initial software architecture work in order to sketch some of the important aspects of an architecture for the future Albert Hall booking system, to address its current limitations. Design and capture some **architectural structures** using some C4 views and make some explicit **architectural decisions** and record them.

When considering your architectural structures, only create views that you think are valuable for capturing, understanding and communicating your architecture. If you need to communicate something that C4 doesn't cover then create another artefact (a diagram or something else) to communicate it … but make sure you define any notation you use clearly!

You may have noticed that none of the approaches are dogmatic about which notation you should use, so choose one or more notations intentionally and use them consistently. Use some sort of shared work area such as a whiteboard (and photograph it to save it), a Miro board, or some shared LucidChart diagrams. Spend your time thinking about content rather than trying to achieve perfect presentation. Content, consistency and correctness are all more important than notational purity, straight lines, aligned shapes and attractive fonts. That said, it is much easier to work with something fairly neat and tidy than a complete mess.

Outputs

When you have spent a couple of hours working on the model, spend about another 60 minutes creating your outputs.

Create 3 or 4 graphically-oriented presentation slides, plus a cover slide clearly indicating your group members. You can use any technology you like to create the slides.

On the slides answer the three questions below. If relevant, illustrate your answers with examples from your Albert Hall booking system architecture.

- Is software architecture an important activity? Why?
- What do you think the most important concerns of software architecture are? Why?
- What was the most important architectural decision you made in the exercise? Why? What are its implications?

It is quite difficult to answer these questions in three legible slides so you need to think carefully about how to answer the question and represent your answer. Get straight to the point! If you have diagrams, make sure you include a notation key somewhere in the material (you can put this on a separate slide if needed).

Upload a representation of your architectural artefacts and your slides as a PDF by **19:00 on Friday 20th October 2023**.

Be prepared to present your slides or part of your architecture in class, as a short 10 minute presentation.

Assessment

The submitted materials will be marked as follows:

- 0 - did not complete the exercise.
- 1 - minimal materials that do not convincingly complete the exercise.
- 3 - good materials that address the main points of the exercise without going beyond the basics of what has been asked for.
- 5 - excellent materials that answer the questions in insightful ways and are very well presented.