Diagramming in Software Engineering

SE Radio episode 566

Authors

Raúl Mera Soto E-mail: UO287827@uniovi.es

Gonzalo Alonso Fernández E-mail: UO282104@uniovi.es

Laura Gómez Menéndez E-mail: UO275725@uniovi.es

Daniel Sinne Argüelles E-mail: UO282500@uniovi.es

Introduction

Ashley Peacock author of “Creating software with modern diagramming techniques” gives his opinion about the importance of diagramming in software engineering. Ashley has a very positive opinion about diagrams, considering their power lies in making information easy to digest. He emphasizes that, in software engineering, diagrams are essential for documenting architectures and explaining complex concepts.

Diagrams are particularly useful in meetings and presentations, making information more understandable. Regarding when to use diagrams in the software life cycle, they can be applied at various stages, such as architecture documentation, proposals, and in situations where visual representation enhances understanding.

It is crucial to distinguish between two types of diagrams: short-lived diagrams and long-lived diagrams. Short-lived diagrams are very useful when you need to understand something you are working on or explain a complex topic to a colleague. These diagrams typically last a few hours or a day and are usually class diagrams or sequence diagrams. Then, there are long-lived diagrams, which add value throughout the entire project, which typically spans many years. The most common long-lived diagrams are domain model diagrams and C4 models. Typically, these diagrams are related to the project's architecture and domain models. It is important to have long-lived diagrams from the beginning of the project life. These diagrams help the entire development team understand the project and know where to go next.

It is also interesting to know which diagramming tools to use to facilitate the creation and maintenance of these types of diagrams; an example of this could be "Mermaid."

Diagramming Tools

Draw.io

Draw.io (now diagrams.net) is a versatile online diagramming tool that allows users to create various types of diagrams, charts, and visual representations effortlessly. It has an easy-to-use drag-and-drop interface, which makes it a perfect tool for quickly creating diagrams. However, this interface can also make it very time-consuming to go back and change anything. It can also be stored in the cloud, making it an option for collaborative work.

PlantUML

PlantUML is a text-based diagramming tool for creating various diagrams. It simplifies the process with an easy syntax, generating diagrams from plain text. It was one of the first tools to introduce the idea of diagrams as code. This text-based approach makes it easy for anyone to quickly understand the diagram and make changes. Because of this it is a good option for cooperative work.

Miro

Miro is a tool that allows us to have a digital whiteboard. It’s not intended for diagramming, although it provides the tools to do so. It’s more useful for a team to perform brainstorming or creating sketches.
It’s commonly used when the members of a team need to work remotely. Its main advantage is that it allows everyone to interact simultaneously with the whiteboard and make contributions.

UML Lab

UML Lab is a modelling tool that can be used to support model driven development. It's integrated with Eclipse and can be installed as an extension. This tool does not only allow to draw and create diagrams from your IDE, but also synchronizes them with your code. In this way, changes on the diagram will be reflected on the code and vice versa.
That makes it easy to use even if we have already written part of the code, we do not need to worry about updating the diagrams since they can be automatically generated.

Mermaid

Mermaid is a powerful yet accessible tool that lets you create diagrams and visualizations using text and code. Like PlantUML, it implements the idea of diagrams as code. It does so by rendering text definitions to create and modify various diagrams dynamically. It is JavaScript based and Markdown-inspired, making it easy to use for those already familiar with these languages.
Mermaid Live Editor also allows non-programmers to easily create detailed diagrams.

All of this allows Mermaid to simplify the process of diagramming by creating easily modifiable diagrams and, therefore, it helps keep documentation up to date with developments, preventing Doc-Rot.

C4 Model

C4 model is a way to model the software architecture was created by Simon Brown.

The name C4 comes from four diagrams:

- System Context Diagram: it is non-technical: no protocols, no databases. It describes the business interactions between the different systems. This is super powerful for showing what we are doing to non-technical people as product managers or stakeholders in a more digestible way, to understand what we are building easily.
- Container View: this one relates to the technical layer, for other people like engineers that want more information. It offers a high-level depiction of the software system's architecture, emphasizing the containers hosting its components. It identifies and visualizes containers, illustrates their interactions and dependencies…
- Component Diagram: it depicts the high-level structure of software components and their interactions and starts defining the system boundary where the major components are identified. Also, relationships between components are shown to illustrate how they interact.
- Code Diagram: It provides a detailed view of the code structure within a specific component or container. It visualizes modules, their interfaces, dependencies… Optionally, it can include implementation details such as methods and attributes for deeper insights into the code structure.