Concept/Topic: *Context, Containers, Components, Classes (C4)*

Context

UML is perceived as heavyweight by some thought leaders and practitioners (but also successfully and gratefully applied by others). The lean and agile movements emphasizes the importance of avoiding waste and focusing on essentials. See for instance the agile modeling practice *iteration modeling*[1].

Definition(s)

In order to reach agile developer communities and make them adopt architecting and visualization practices, S. Brown created a simplified approach to design modeling, which he calls the *C4 model*. C4 stands for Context, Containers, Components, Classes; these are the four main diagram types he suggests. They stand in a hierarchical relationship, so one can zoom into elements on one level and then see its contained elements (for instance, clicking on a container displays the components in this container). The three core parts of the C4 model are:

- *Context diagram*: "A System Context diagram is a good starting point for diagramming and documenting a software system, allowing you to step back and see the big picture. Draw a diagram showing your system as a box in the centre, surrounded by its users and the other systems that it interacts with. Detail isn't important here as this is your zoomed out view showing a big picture of the system landscape. The focus should be on people (actors, roles, personas, etc) and software systems rather than technologies, protocols and other low-level details. It's the sort of diagram that you could show to non-technical people."
- *Container diagram*: "Once you understand how your system fits in to the overall IT environment, a really useful next step is to zoom-in to the system boundary with a Container diagram. A"container" is something like a web application, desktop application, mobile app, database, file system, etc. Essentially, a container is a separately deployable unit that executes code or stores data. The Container diagram shows the high-level shape of the software architecture and how responsibilities are distributed across it. It also shows the major technology choices and how the containers communicate with one another. It's a simple, high-level technology focussed diagram that is useful for software developers and support/operations staff alike. […]
- *Component diagram*: "Next you can zoom in and decompose each container further to identify the major structural building blocks and their interactions. The Component diagram shows how a container is made up of a number of"components", what each of those components are, their responsibilities and the technology/implementation details.

Context diagram and contaoner diagram target technical and non-technical people, inside and outside of the immediate software development team. The intended audience of component and class diagram are technical people within the software development team.

S. Brown further explains the concepts of container and component like this: A *container* represents something that hosts code or data, so essentially it is a context or boundary inside which some code is executed or some data is stored. Each container is separately deployable:

- Server-side web application: A Java EE web application running on Apache Tomcat, an ASP.NET MVC application running on Microsoft IIS, a Ruby on Rails application running on WEBrick, a Node.js application, etc.

[1] http://agilemodeling.com/essays/iterationModeling.htm

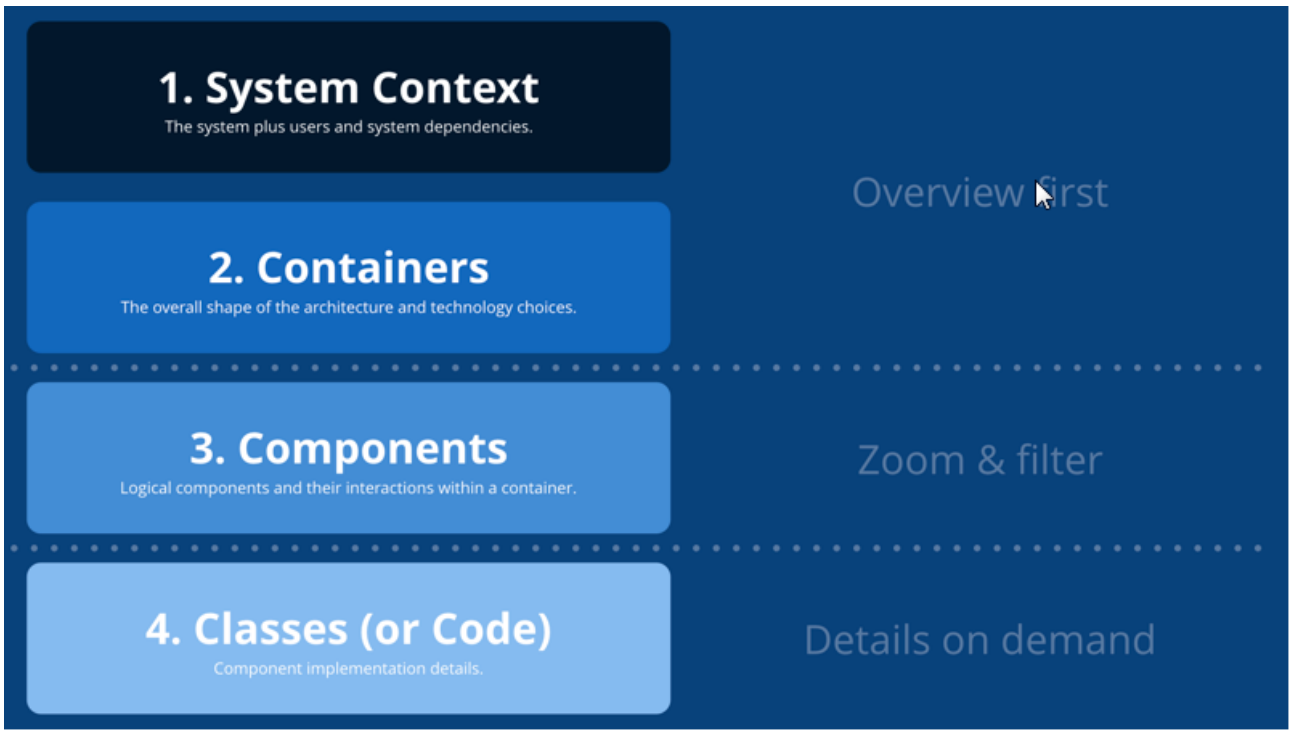

Figure 1: *S. Brown's C4 Model for Architecture Visualization*

- Client-side web application: A JavaScript application running in a web browser using Angular, Backbone.JS, jQuery, etc.).
- Client-side desktop application: A Windows desktop application written using WPF, an OS X desktop application written using Objective-C, a cross-platform desktop application written using JavaFX, etc.
- Mobile app: An Apple iOS app, an Android app, a Microsoft Windows Phone app, etc.
- Server-side console application: A standalone (e.g. "public static void main") application, a batch process, etc.
- Microservice: A single microservice, hosted in anything from a traditional web server to something like Spring Boot, Dropwizard, etc.
- Database: A schema or database in a relational database management system, document store, graph database, etc such as MySQL, Microsoft SQL Server, Oracle Database, MongoDB, Riak, Cassandra, Neo4j, etc.
- Blob or content store: A blob store (e.g. Amazon S3, Microsoft Azure Blob Storage, etc.) or content delivery network (e.g. Akamai, Amazon CloudFront, etc.).
- File system: A full local file system or a portion of a larger networked file system (e.g. SAN, NAS, etc).
- Shell script: A single shell script written in Bash, etc."

"The word *component* is a hugely overloaded term in the software development industry, but in this context a component is simply a grouping of related functionality encapsulated behind a well-defined interface. If you're using a language like Java or C#, the simplest way to think of a component is that it's a collection of implementation classes behind an interface. Aspects such as how those components are packaged (e.g. one component vs many components per JAR file, DLL, shared library, etc) is a separate and orthogonal concern." (source: https://c4model.com/)

Extensions

Two more diagrams have been added recently, capturing deployment details and architecture dynamics; a variation of the context diagram called enterprise context diagram is now also part of the model (it remains to be seen if C4 eventually ends up with an expressiveness and complexity similar to UML).

Example(s)

See lecture and exercise 3 for the time being, as well as online resources.

Application of the Concept in Products and Projects

Usage of Concept/Topic

The concept has been promoted by S. Brown through many well-received presentations in recent years. Use "coding the architecture" as an additional search keyword. Other architect have picked the notion up, see for instance this article[2].

Tool Support

Structurizr is S. Brown's implementations of his concepts. Additonal ones have been provided by the community. See bottom of the C4 home page[3]. Structurizr is an Architecture Description Language (ADL) that comes as a Java and C# API with supporting tools such as Structurizr Express[4]. The latter Web-based tool supports a subset of C4.

Tips and Tricks

- Consider agile whiteboard modeling before investing in tooling licenses and skills.
- Critically review what is really new, and map the C4 concepts to notations you already know (for instance, UML). Distinguish sales messages from facts.
- In all four views, do specify statics and dynamics. For dynamics, C4 suggests collaboration diagrams but you can also use UML sequence diagrams as *component interaction diagrams*, for example created with this online tool: https://www.websequencediagrams.com/

More Information

Related Topics and Concepts

- Methods (for instance, C4 models can become part of the Architecture Notebook[5] artifact in OpenUP)
- Agile architecting
- Component modeling

[2] https://www.codeproject.com/Articles/1167140/The-C-Software-Architecture-Model%3E
[3] https://c4model.com/
[4] https://structurizr.com/express
[5] http://epf.eclipse.org/wikis/openup/practice.tech.evolutionary_arch.base/workproducts/architecture_notebook_9BB92433.html

Miscellaneous Links:

You can find introductory material and experience reports about C4 here:

- C4 homepage[6]
- Article by S. Brown[7]
- Another article[8]

The concepts are also explained in this article (with examples)[9].

More information on models and modeling is available via the IFS website Method Selection and Tailoring Guide[10].

[6] https://c4model.com/
[7] https://www.voxxed.com/2014/10/simple-sketches-for-diagramming-your-software-architecture/
[8] http://www.methodsandtools.com/archive/softwarearchitecturesketches.php
[9] http://www.qappdesign.com/the-c4-software-architecture-model/
[10] https://www.ifs.hsr.ch/index.php?id=13195&L=4