WIKIPEDIA

# Hexagonal architecture (software)

The **hexagonal architecture**, or **ports and adapters architecture**, is an architectural pattern used in software design. It aims at creating loosely coupled application components that can be easily connected to their software environment by means of ports and adapters. This makes components exchangeable at any level and facilitates test automation.[1]

Contents

## Origin

The hexagonal architecture was invented by Alistair Cockburn in an attempt to avoid known structural pitfalls in object-oriented software design, such as undesired dependencies between layers and contamination of user interface code with business logic, and published in 2005.[2]

The term "hexagonal" comes from the graphical conventions that shows the application component like a hexagonal cell. The purpose was not to suggest that there would be six borders/ports, but to leave enough space to represent the different interfaces needed between the component and the external world.[1]

## Principle

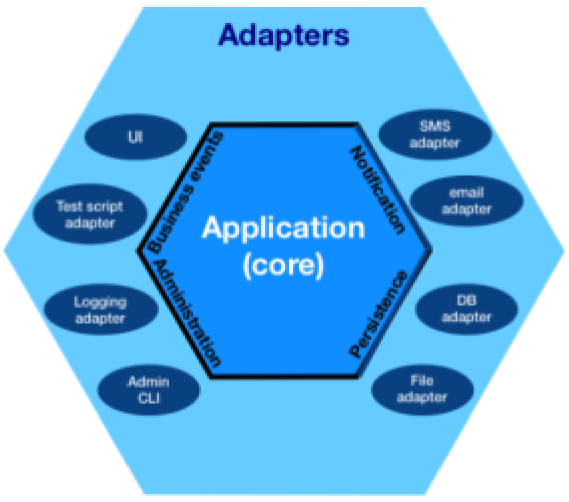
The hexagonal architecture divides a system into several loosely-coupled interchangeable components, such as the application core, the database, the user interface, test scripts and interfaces with other systems. This approach is an alternative to the traditional layered architecture.

Each component is connected to the others through a number of exposed "ports". Communication through these ports follow a given protocol depending on their purpose. Ports and protocols define an abstract API that can be implemented by any suitable technical means (e.g. method invocation in an object-oriented language, remote procedure calls, or Web services).

The granularity of the ports and their number is not constrained:

- a single port could in some case be sufficient (e.g. in the case of a simple service consumer) ;
- typically, there are ports for event sources (user interface, automatic feeding), notifications (outgoing notifications), database (in order to interface the component with any suitable DBMS), and administration (for controlling the component);
- in an extreme case, there could be a different port for every use case, if needed.

Adapters are the glue between components and the outside world. They tailor the exchanges between the external world and the ports that represent the requirements of the inside of the application component. There can be several adapters for one port, for example if data can be provided by a user through a GUI or a command-line interface, by an automated data source, or by test scripts.



Example of hexagonal architecture

## Criticism

The term "hexagonal" implies that there are 6 parts to the concept, whereas there are only 4 key areas. The term’s usage comes from the graphical conventions that shows the application component like a hexagonal cell. The purpose was not to suggest that there would be six borders/ports, but to leave enough space to represent the different interfaces needed between the component and the external world.[3]

According to Martin Fowler, the hexagonal architecture has the benefit of using similarities between presentation layer and data source layer to create symmetric components made of a core surrounded by interfaces, but with the drawback of hiding the inherent asymmetry between a service provider and a service consumer that would better be represented as layers.[4]

## Evolution

According to some authors, the hexagonal architecture is at the origin of the microservices architecture.[5]

## Variants

The onion architecture proposed by Jeffrey Palermo in 2008 is similar to the hexagonal architecture: it also externalizes the infrastructure with proper interfaces to ensure loose coupling between the application and the database.[6] It decomposes further the application core into several concentric rings using inversion of control.[7]

The clean architecture proposed by Robert C. Martin in 2012 combines the principles of the hexagonal architecture, the onion architecture and several other variants; It provides additional levels of detail of the component, which are presented as concentric rings. It isolates adapters and interfaces (user interface, databases, external systems, devices) in the outer rings of the architecture and leaves the inner rings for use cases and entities[8], [9] The clean architecture uses the principle of dependency inversion with the strict rule that dependencies shall only exist between an outer ring to an inner ring and never the contrary.

## See also

- Architecture patterns
- Layer (object-oriented design)
- Composite structure diagram
- Object oriented Analysis and design

## References

1. Alistair, Cockburn (2005-04-01). "Hexagonal architecture" (https://alistair.cockburn.us/hexagonal-architecture/). *alistair.cockburn.us*. Retrieved 2020-11-18.
2. Stenberg, Jan (2014-10-31). "Exploring the Hexagonal Architecture" (https://www.infoq.com/news/2014/10/exploring-hexagonal-architecture/). *InfoQ*. Retrieved 2019-08-12.
3. Alistair, Cockburn (2005-04-01). "Hexagonal architecture" (https://alistair.cockburn.us/hexagonal-architecture/). *alistair.cockburn.us*. Retrieved 2020-11-18.
4. Fowler, Martin (2003). *Patterns of enterprise application architecture*. Addison-Wesley. p. 21. ISBN 0-321-12742-0. OCLC 50292267 (https://www.worldcat.org/oclc/50292267).
5. Rajesh R. V. (2017). *Spring 5.0 microservices : build scalable microservices with Reactive Streams, Spring Boot, Docker, and Mesos* (Second ed.). Packt Publishing. pp. 13–14. ISBN 978-1-78712-051-8. OCLC 999610958 (https://www.worldcat.org/oclc/999610958).
6. Jeffrey, Palermo (2008-07-29). "The Onion Architecture : part 1" (https://jeffreypalermo.com/2008/07/the-onion-architecture-part-1/). *Programming with Palermo*. Retrieved 2019-08-12.
7. Chatekar, Suhas (2015). *Learning NHibernate 4 : explore the full potential of NHibernate to build robust data access code*. Packt Publishing. pp. 249–250. ISBN 978-1-78439-206-2. OCLC 937787252 (https://www.worldcat.org/oclc/937787252).
8. Martin, Robert, C. (2012-08-12). "The Clean architecture | Clean Coder Blog" (https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html). *blog.cleancoder.com*. Retrieved 2019-08-12.
9. Martin, Robert C. (2017). *Clean architecture : a craftsman's guide to software structure and design*. Prentice Hall. ISBN 978-0-13-449416-6. OCLC 1004983973 (https://www.worldcat.org/oclc/1004983973).

Retrieved from "https://en.wikipedia.org/w/index.php?title=Hexagonal_architecture_(software)&oldid=1035644146"