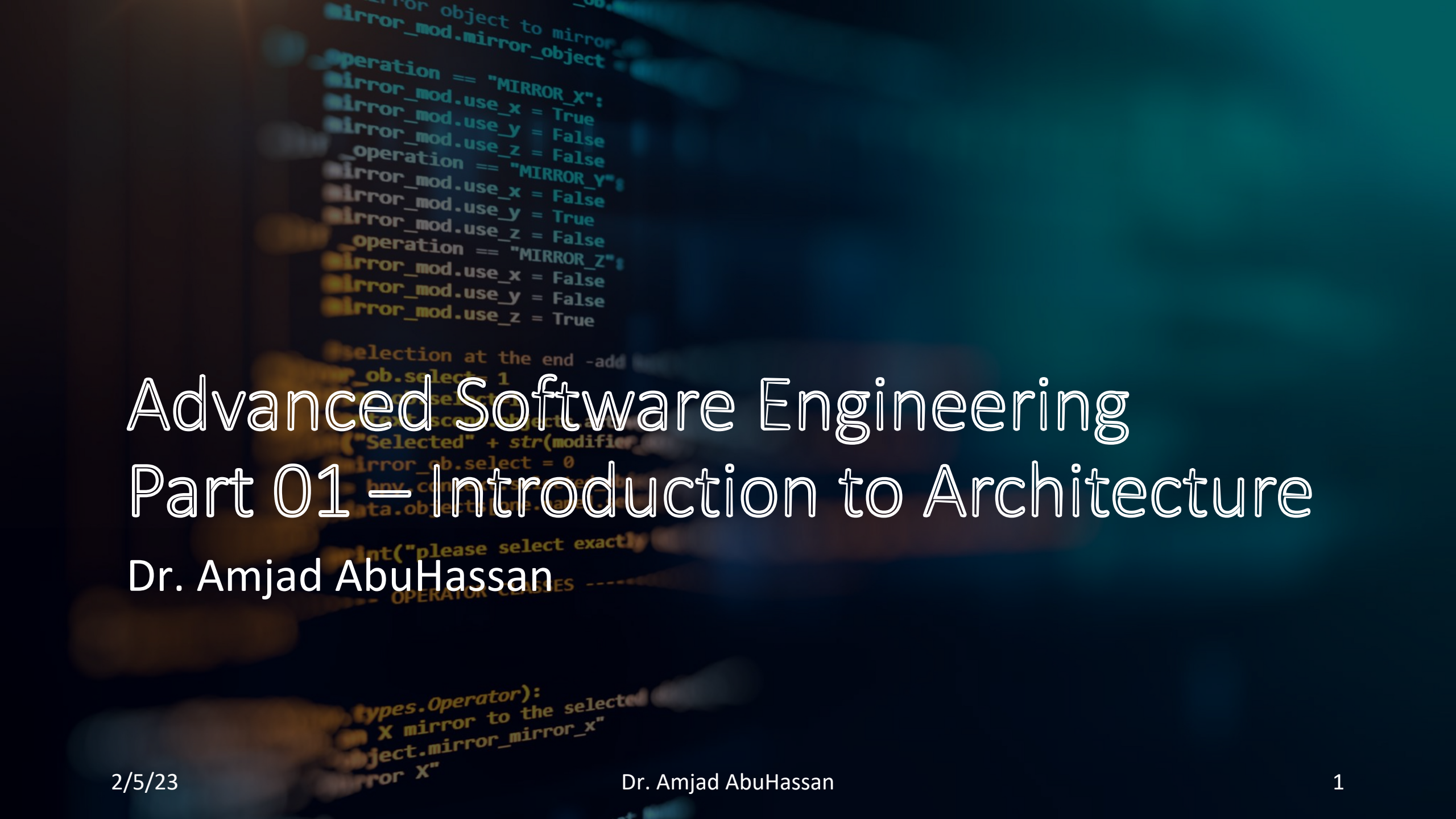


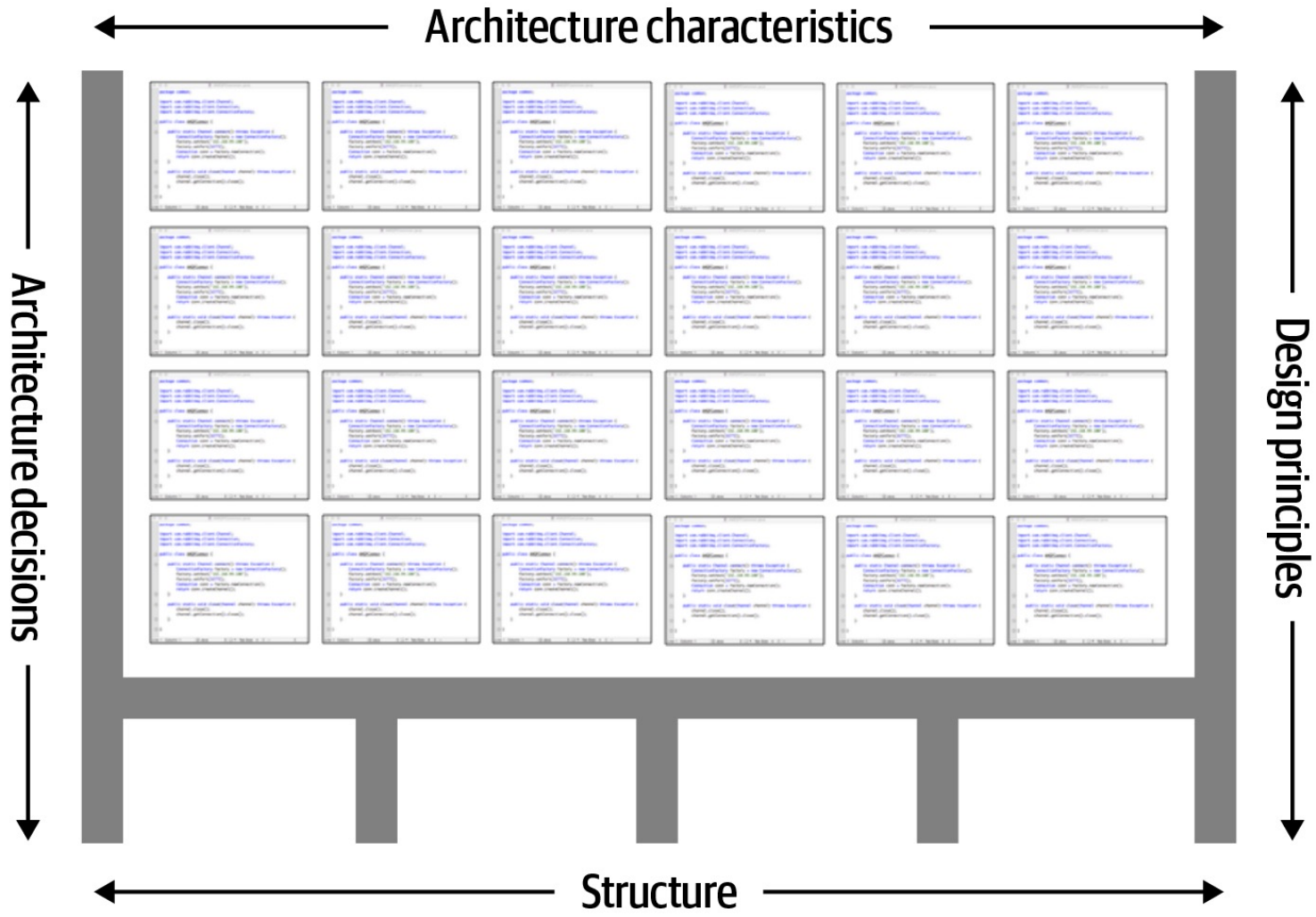
Advanced Software Engineering
Part 01 – Introduction to Architecture

Dr. Amjad AbuHassan

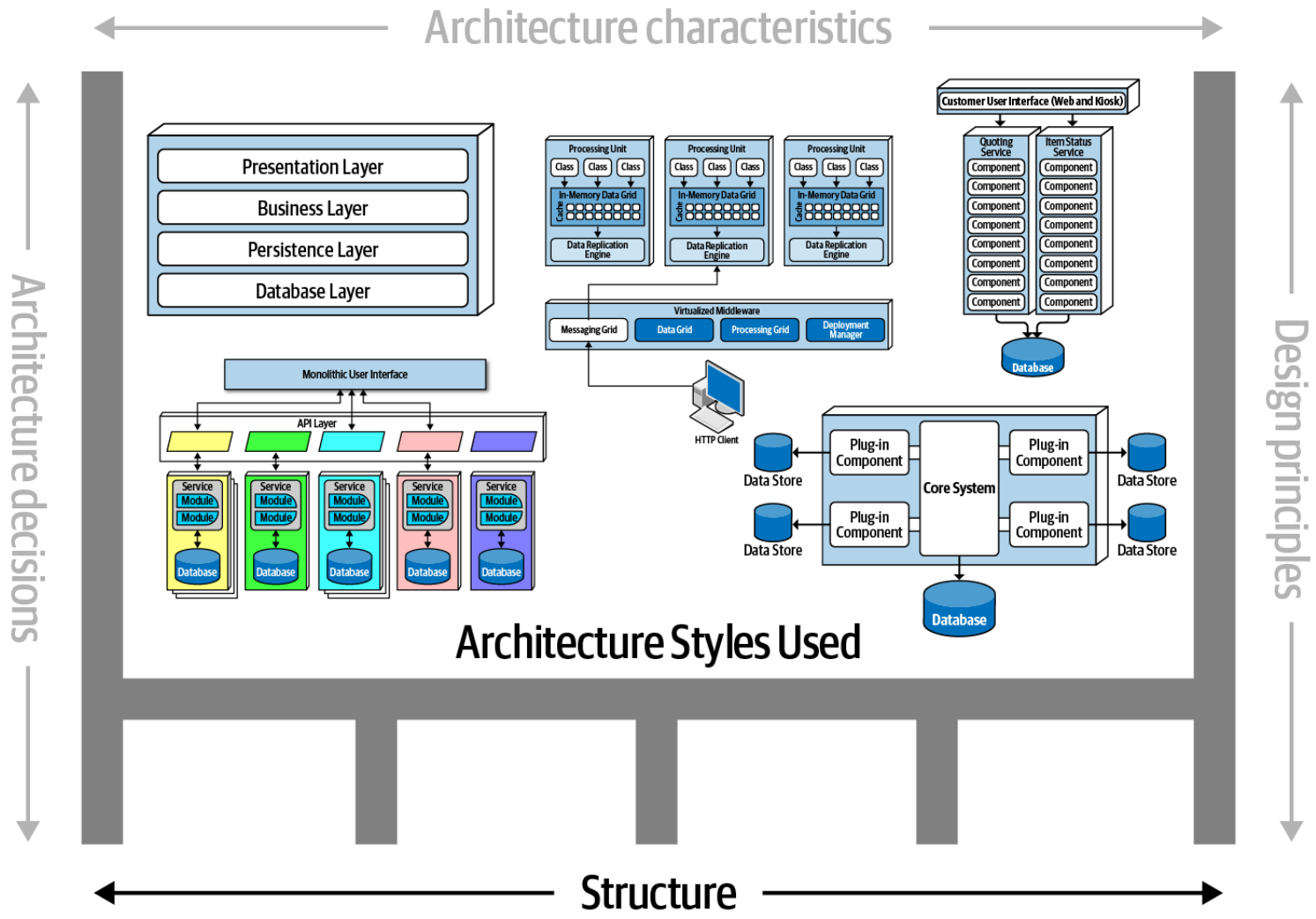
Outline

- Defining Software Architecture
- Architectural Thinking
- Modularity

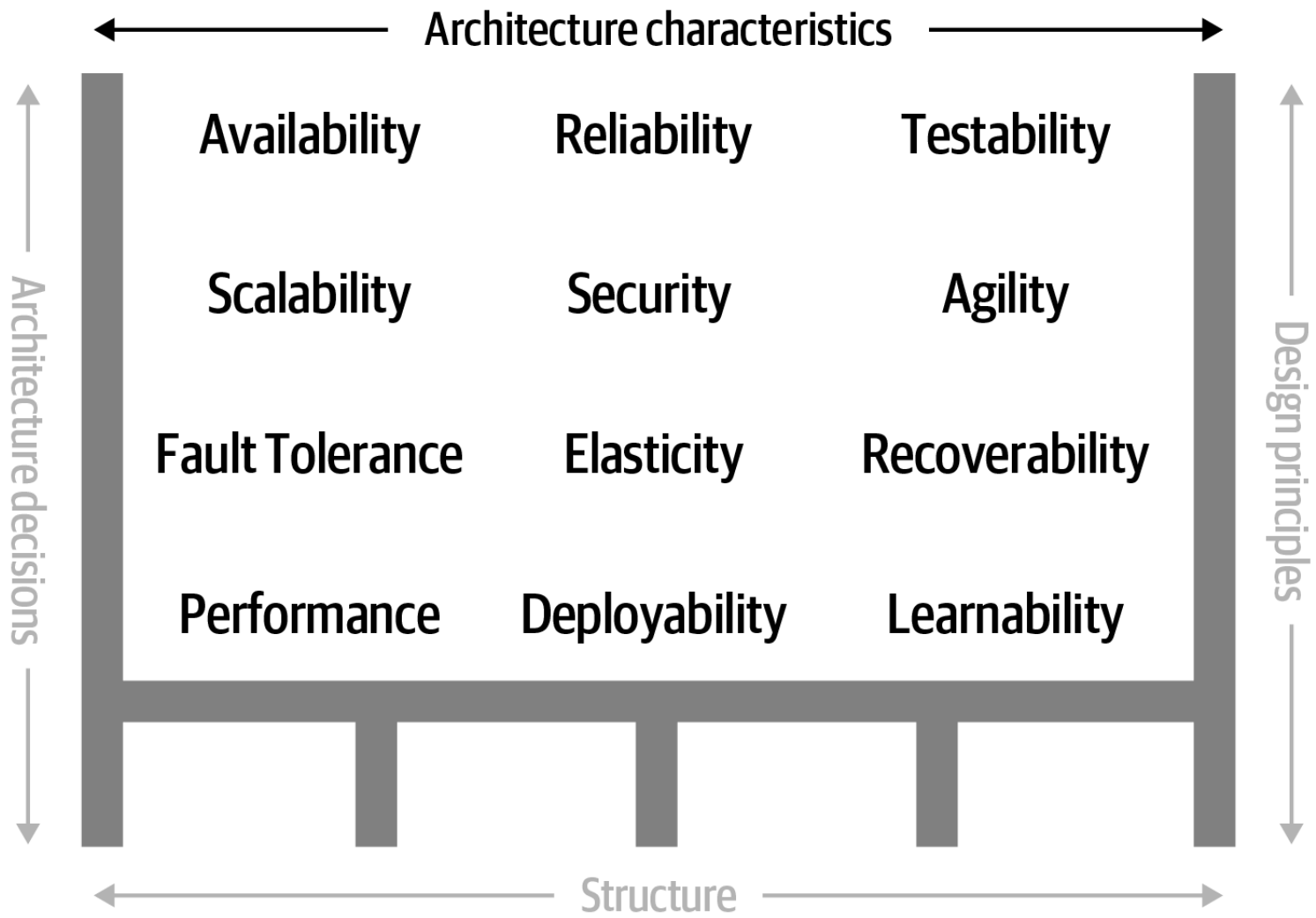
Defining Software Architecture



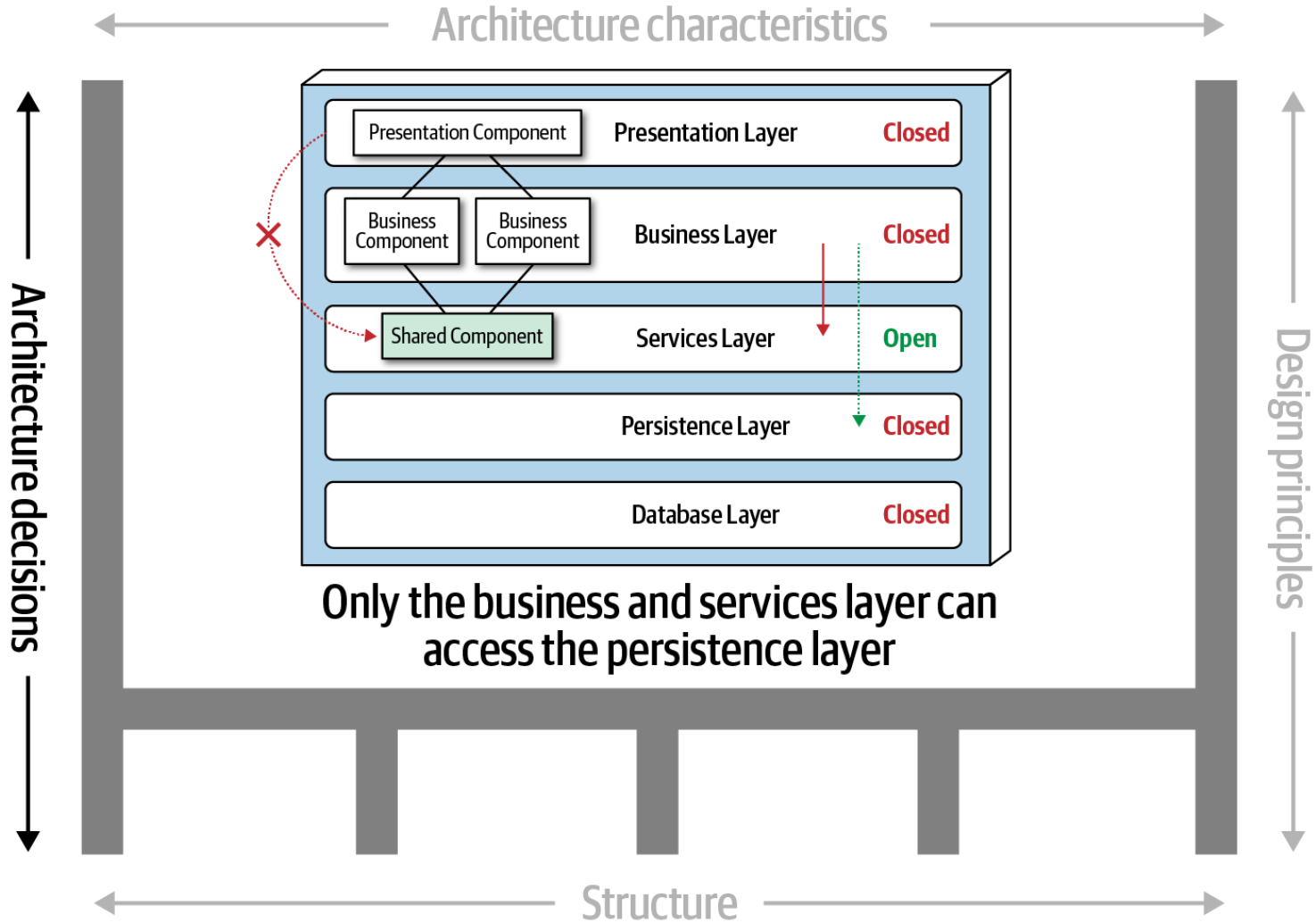
Software Structure



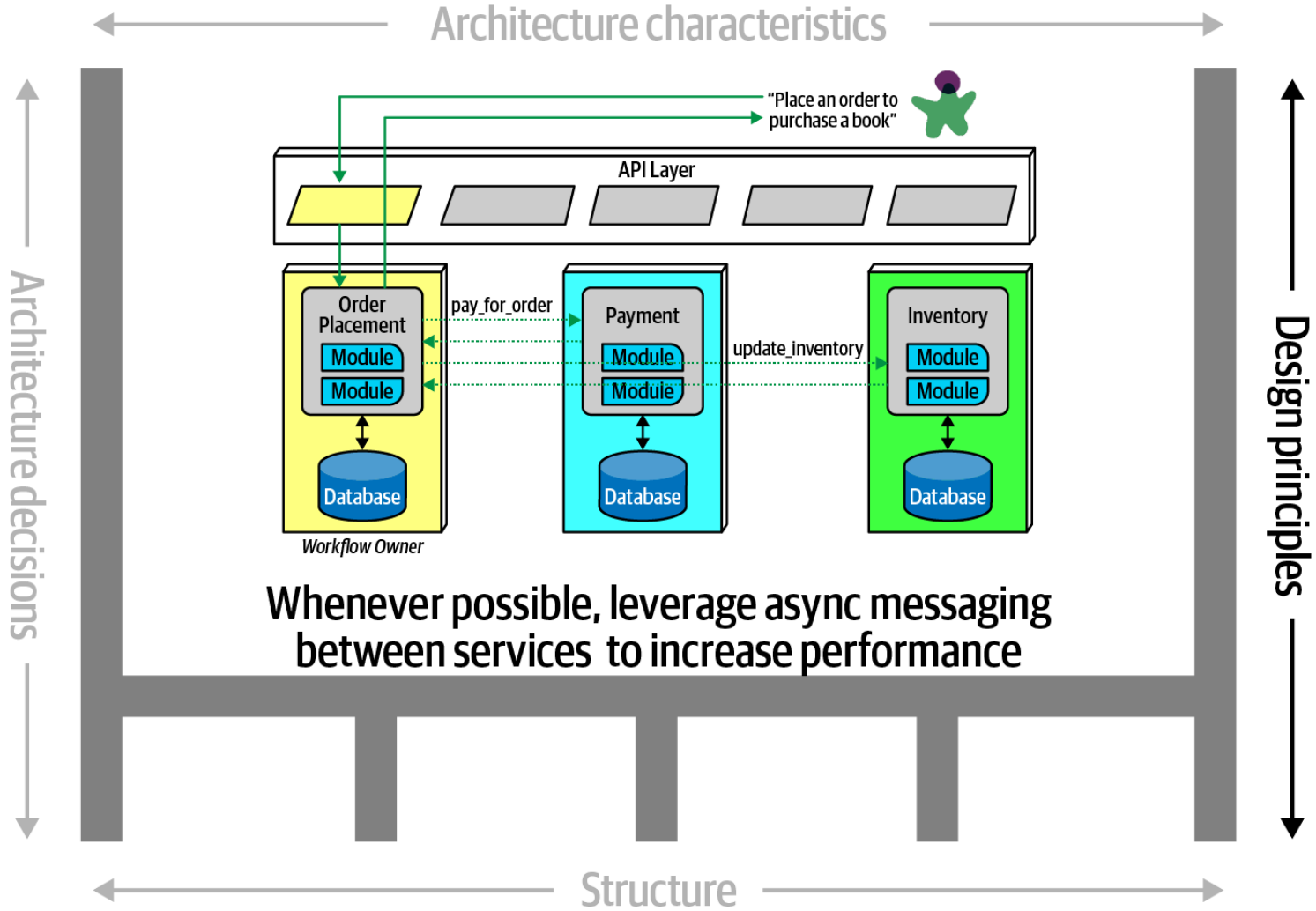
Architecture Characteristics



Architecture Decisions



Design Principles



Laws of Software Architecture

- First Law of Software Architecture:

  Everything in software architecture is a trade-off.

- Second Law of Software Architecture:

  Why is more important than how.

Analyzing Trade-Offs

- Architecture is the stuff you can't Google.
- Everything in architecture is a trade-off, which is why the famous answer to every architecture question in the universe is "it depends."
- There are no right or wrong answers in architecture—only trade-offs.
- Programmers know the benefits of everything and the trade-offs of nothing. Architects need to understand both.

Modularity

- We use modularity to describe a logical grouping of related code, which could be a group of classes in an object-oriented language or functions in a structured or functional language.

- Measuring Modularity
  - Cohesion
  - Coupling

Cohesion

- Cohesion refers to what extent the parts of a module should be contained within the same module.
- Ideally, a cohesive module is one where all the parts should be packaged together, because breaking them into smaller pieces would require coupling the parts together via calls between modules to achieve useful results.

Attempting to divide a cohesive module would only result in increased coupling and decreased readability.

~ Larry Constantine

Example

- For example, consider this module definition:

```
Customer Maintenance
add customer
update customer
get customer
notify customer
get customer orders
cancel customer orders
```

- Should the last two entries reside in this module or should the developer create two separate modules

Example cont.

Customer Maintenance

- add customer
- update customer
- get customer
- notify customer

Order Maintenance

- get customer orders
- cancel customer orders

Which is the correct structure? As always, it depends:

Coupling

- Afferent coupling measures the number of incoming connections to a code artifact (component, class, function, and so on).

- Efferent coupling measures the outgoing connections to other code artifacts.