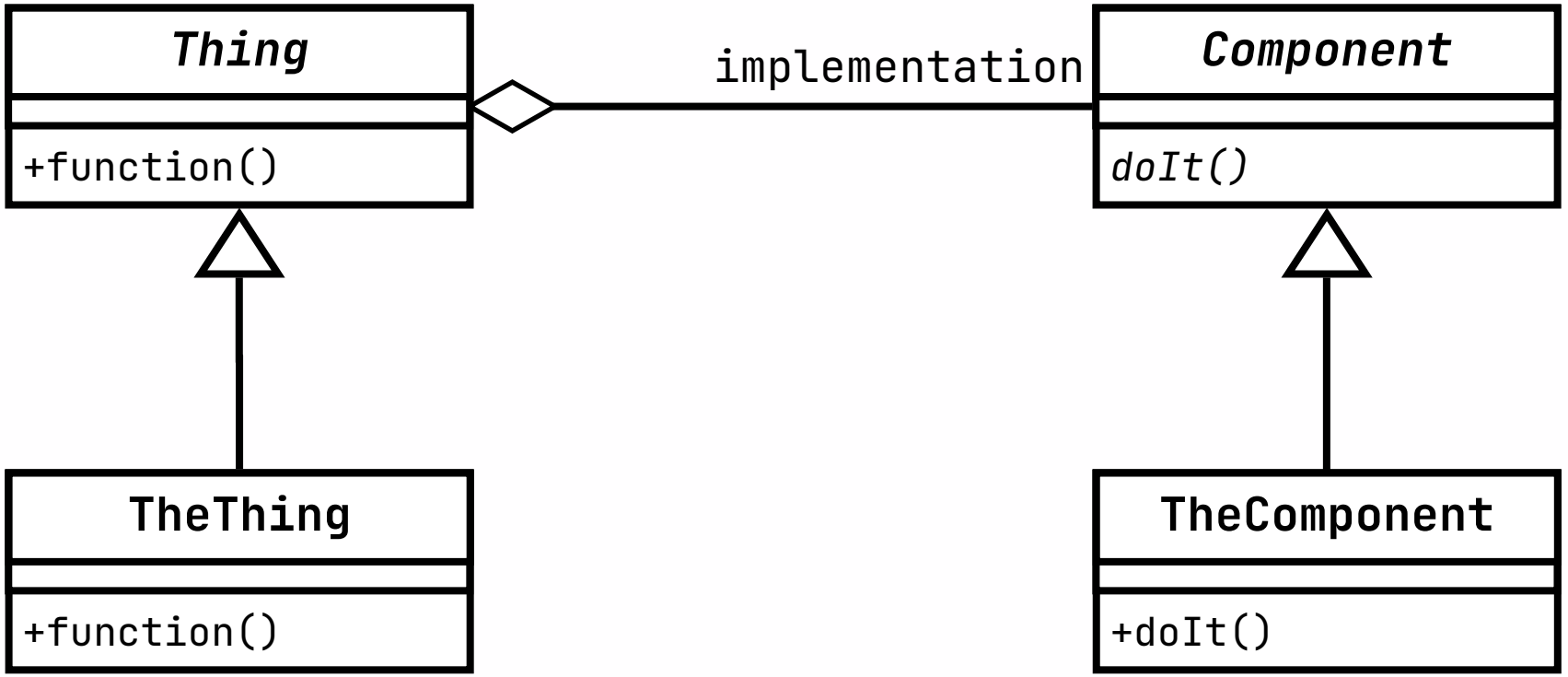
Design Patterns

Software Systems — Programming — 5M1

Dr. Vadim Zaytsev aka @grammarware, November/December 2020

UNIVERSITY OF TWENTE. | Formal Methods & Tools

Example

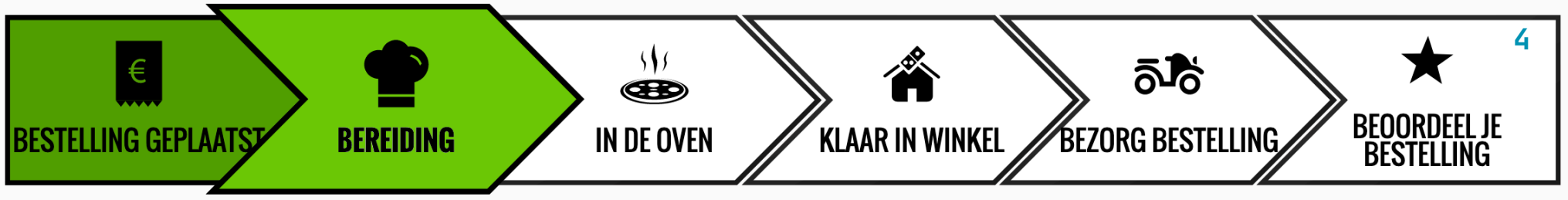

Thing
+function()
implementation
Component
doIt()
TheThing
+function()
TheComponent
+doIt()

Strategy

- Context + Strategy
- Algorithm families:
  - access database
  - break stream into chunks
  - optimise generated code
  - validate user input

GoF, *Design Patterns*, p. 315.

State



- Context + State
- Anything resembling a state machine:
  - ordering pizza
  - “active tool” in editors
  - communication protocols (TCP)
  - coffee machine

GoF, *Design Patterns*, p. 305.

Bridge

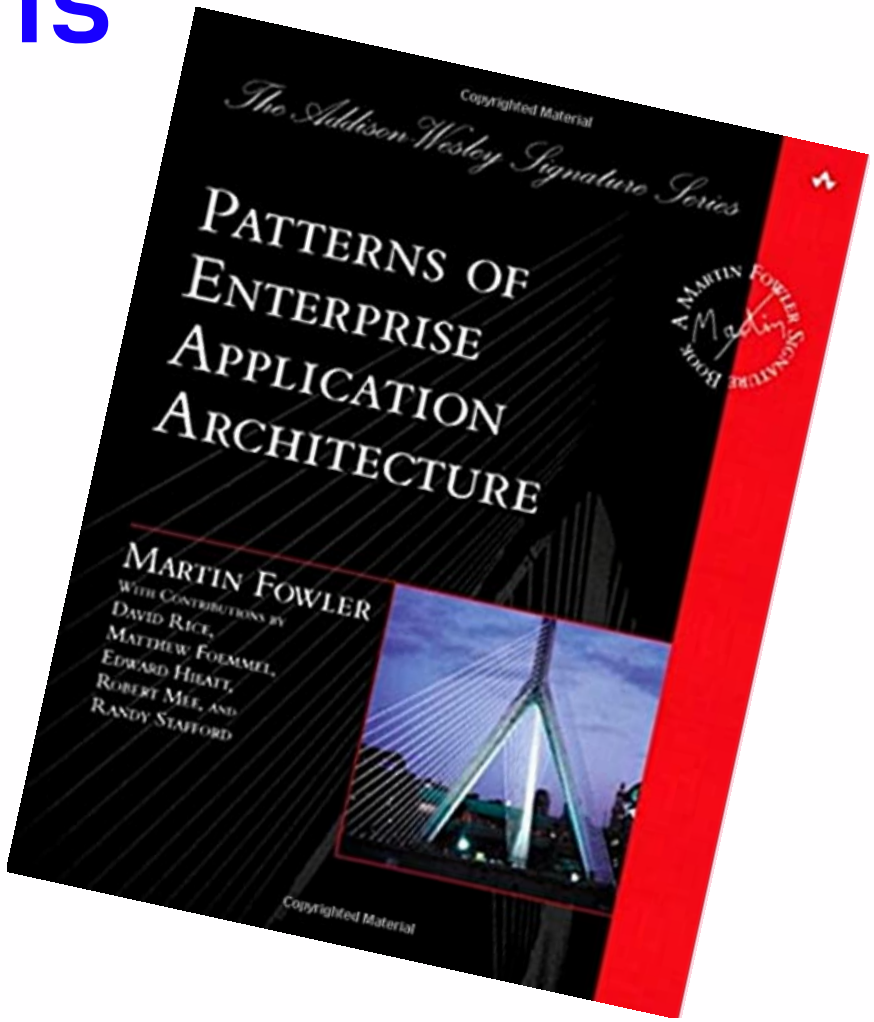
- Abstraction + Implementation
- Shape ◁— Triangle, Square, …
  Colour ◁— Red, Blue, …
- Window ◁— IconWindow, PopupWindow, …
  WindowImp ◁— XWindowImp, PMWindowImp,…
- Device ◁— TV, PS, …
  Remote ◁— BasicRemote,AdvancedRemote,…

GoF, *Design Patterns*, p. 151.

Design Patterns

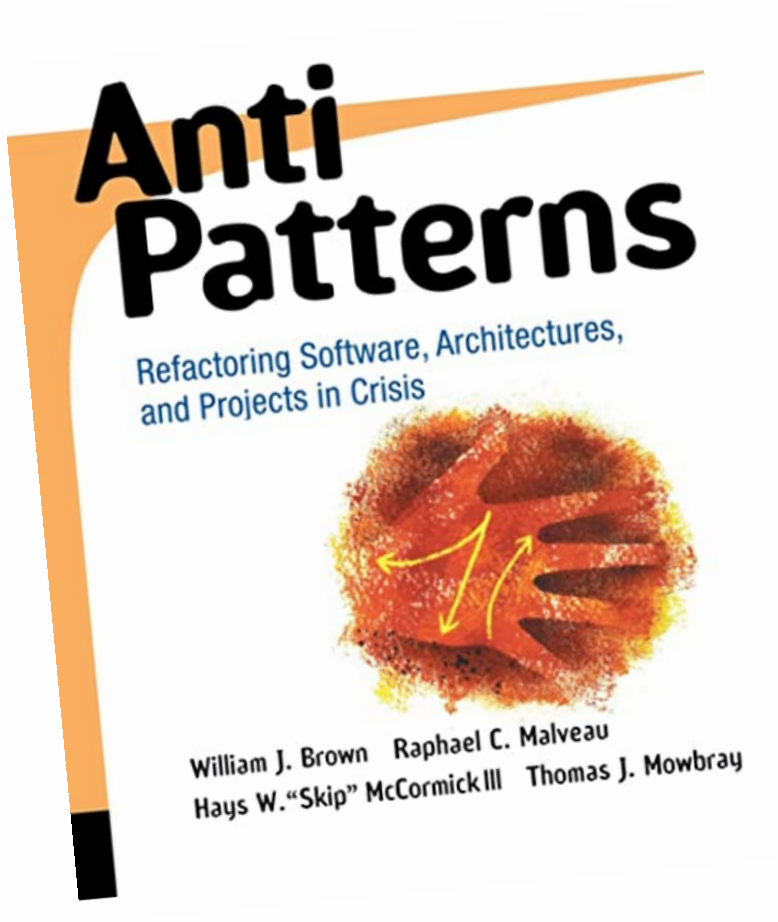
- Structural
  - about composition of classes/objects
- Behavioural
  - about interaction & responsibility
- Creational
  - about object creation

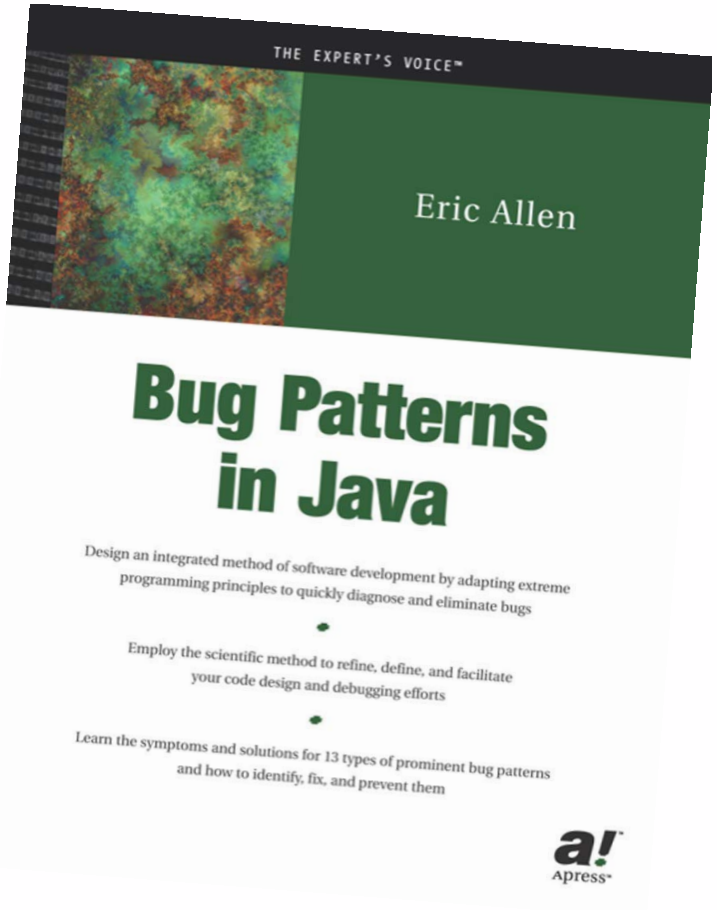
Design Patterns



Design Patterns (1994), *Patterns of EAA* (2002)

Antipatterns





AntiPatterns (1998), *Bug Patterns* (2002)

How to Choose a Pattern?

- Formulate the problem well
- Know the basic catalogue
- Make sure the intent fits
- Study alternatives
- Keep redesign in mind
- Do not overengineer
- Isolate the client
- Think

Conclusion

- **Design pattern**
  - is a named abstraction
  - for a recurring solution
  - to a particular design problem
- Represent design knowledge (not diagrams)
- 23 in the GoF book, others elsewhere
  - name, problem, solution, consequences
- *Patterns are discovered, not invented*