Typestate Pattern in Rust

With Fluent Constructor and State Machine Examples

David Wallace Croft, M.Sc.

Presented to the
Indy Rust User Meetup
2024 Sep 04 Wed

Design Patterns

- Borrowed from a book on building architecture
- Adopted by software architects
  - Gamma, et al., "Design Patterns", 1994
  - Known as the "Gang of Four Book"
- A reusable pattern that fits a type of situation
  - Problem and solution
  - Customized as needed
- Quickly communicate design ideas
  - Using just the name of the pattern

Antipatterns

- Sub-optimal design patterns
  - Used frequently enough to be named
- Should generally be avoided
  - Disadvantages outweigh the advantages

Builder Pattern

- Problem
  - Make a recipe to assemble a complex object
  - Enable swapping out implementations
- Solution
  - Client instantiates a ConcreteBuilder
  - Client passes ConcreteBuilder to Director
  - Director operates on an AbstractBuilder
  - Client retrieves Product from ConcreteBuilder

Named Arguments

- A potential run-time error
  - my_function(height, width)
  - my_function(width, height)
- Named arguments (a.k.a. named parameters)
  - my_function(width => width, height => height)
- Languages with named arguments
  - Ada, C#, Fortran, Kotlin, Python, Ruby, [...]
- Rust
  - https://github.com/rust-lang/rfcs/issues/323

Builder Antipattern

- Work-around for a lack of named arguments
  - let p = Product::builder().a(a).b(b).build();
- Different from Gang of Four book definition
  - To distinguish, I call it a "Fluent Constructor"
- Easily misused resulting in run-time errors
  - Building before all required arguments given
  - Reusing after build when not designed for it
  - Permits invalid argument combinations
  - Breaks when arguments added to constructor

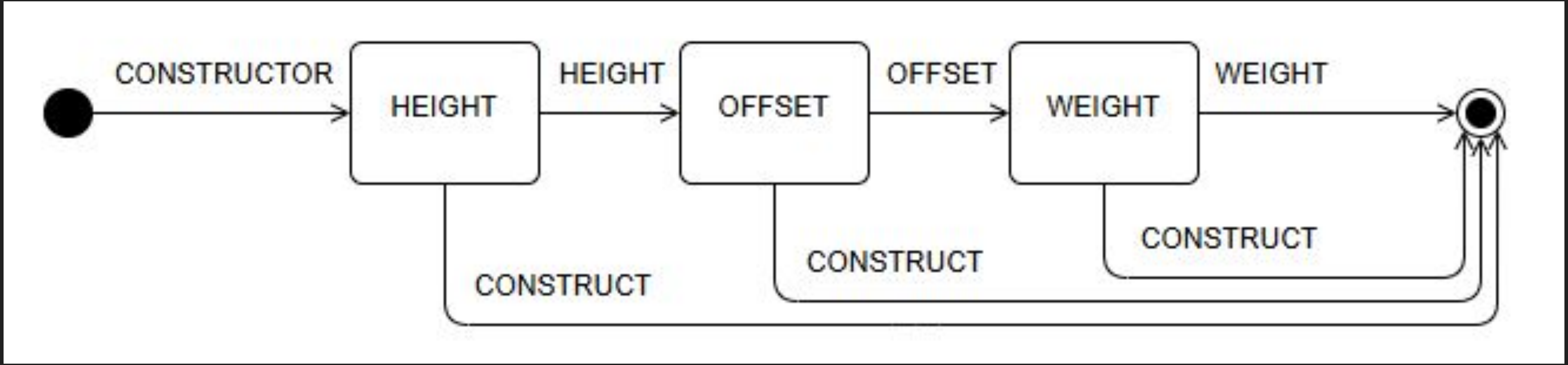
Typestate Pattern

- Problem
  - Permit state transitions only when valid
  - Enforce using static compile-time checks
- Solution
  - Represent the states using typestates (structs)
  - State transition methods are typestate-specific
  - State transition methods consume self

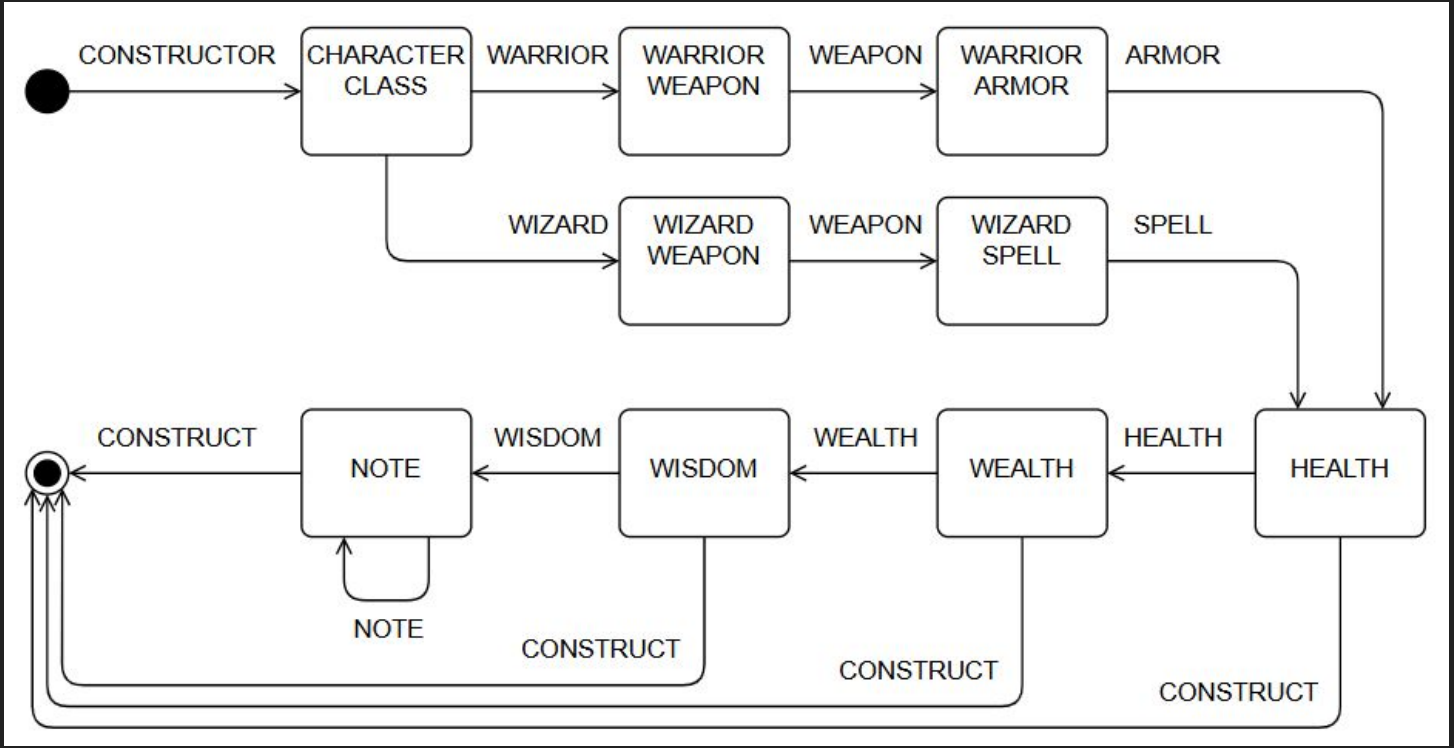
Typestate Fluent Constructor

- A Fluent Constructor that cannot be misused
  - Based on the Typestate Pattern
  - Also called a "Strict Builder"
- Compile-time errors instead of run-time errors
  - Cannot build until all required values provided
  - Prevents invalid argument combinations
  - New arguments require code updates
  - Initial arguments determine next ones allowed
  - Prevents reuse after build

Widget Fluent Constructor



Player Character Fluent Constructor

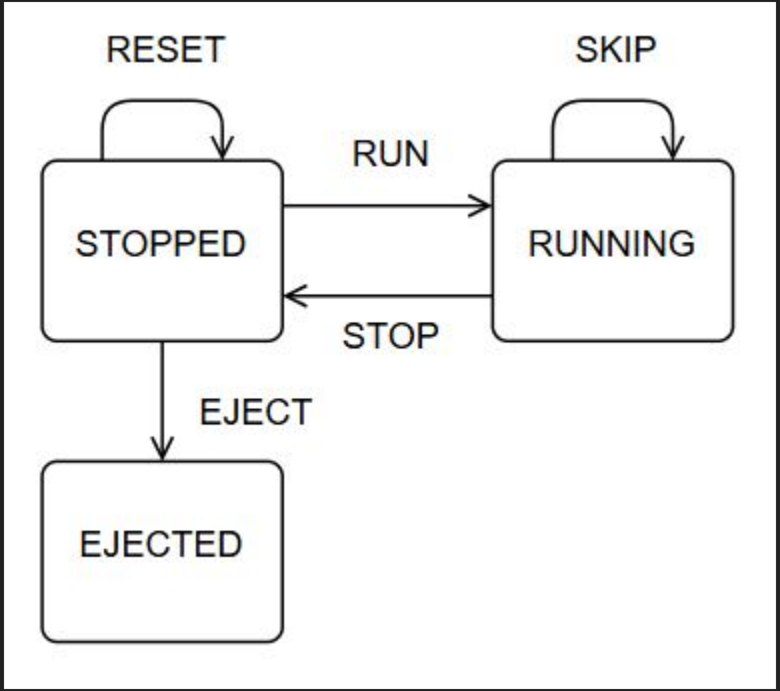


Example Code

- Open source example code on GitHub
  - https://github.com/david-wallace-croft/pattern-typestate

- fluent_constructor_0
  - A basic typestate fluent constructor
- fluent_constructor_1
  - Specifying the state using a generic and phantom data
- fluent_constructor_2
  - A typestate fluent constructor for an external struct
- fluent_constructor_3
  - Diverging and converging chain method paths

State Machine



Typestate State Machine

- State Machine
  - Changes system state upon event triggers
  - Only implements valid state transitions
- How to use asynchronous events with Typestate?
  - Event handling dependent on state value
  - But the typestate is a type, not a value
- Store the typestate in an enum variant field
  - Extract the typestate in an enum matching arm

Example Code

- Open source example code on GitHub
  - https://github.com/david-wallace-croft/pattern-typestate

- state_machine_0
  - Operates on data inside itself
- state_machine_1
  - Operates on data outside itself

Links

- Cliff L. Biffle, "The Typestate Pattern in Rust", 2019-06-05, https://cliffle.com/blog/rust-typestate/
- Eric Smith, "Game Development with Rust and WebAssembly", 2022 Apr, https://www.packtpub.com/en-us/product/game-development-with-rust-and-webassembly-9781801070973/
- Gamma, et al., "Design Patterns: Elements of Reusable Object-Oriented Software" (1E), Addison-Wesley Professional, 1994. https://en.wikipedia.org/wiki/Design_Patterns

Presenter

- David Wallace Croft, M.Sc.
  - https://www.CroftSoft.com/people/david/
- Organizer of the Dallas Rust User Meetup
  - https://www.DallasRust.org/
- Open source Rust projects
  - Animated interactive games and simulations that run in the browser using WebAssembly (Wasm)
  - Single page applications (SPAs) with static pre-rendering and client-side hydration using Dioxus
  - Serverless functions using Amazon Web Services (AWS) Lambda and Fermyon Spin
  - https://www.CroftSoft.com/people/david/research/rust-wasm/

Licenses