Inhaltsverzeichnis

1 Übersicht

In diesem Artikel fasse ich meine Gedanken zu dem Thema Pattern im Kontext der C++ Programmierung und im Kontext der Systementwicklung zusammen. Ziel ist es von einer Systemidee zu einer Software-Architektur zu kommen um dann darauf aufbauend die Implementierung in C++ umzusetzen. Hierbei spielen Pattern eine wichtige Rolle.

Ich habe mir dafür ein Projekt aufgesetzt, anhand dessen ich die einzelnen Abschnitte ausprobieren will[1].

Als Basis für die Einarbeitung nutze ich das bekannte Buch der GoF[2] und die Artikel Serie von Rainer Grimm im C++ Blog von Heise Online[3].

Ich beschränke mich in diesem Artikel auf die wesentlichen Inhalte und vermeide ausschweifende Beschreibungen.

2 Einstieg ins Thema Pattern

Man kann sich dem Thema Pattern aus unterschiedlichen Richtungen nähern, die Grundidee ist aber immer die selbe:

Ich versuche ein System in einzelne Elemente runterzubrechen und diese dann so oft wie möglich in weiteren Projekten wieder zu verwenden.

Die einzelnen Elemente sollen dann über eine Art von Standart-Lösung umgesetzt werden. Dass sind dann die bekannten Pattern.

Ziel ist es somit ein System über einzelne, bereits bekannte Muster aufzubauen um das Rad nicht immer wieder neu zu erfinden.

Da ein System aus unterschiedlichen Elemente besteht, gibt es auf unterschiedlich Sichten auf die einzelnen Elemente. Die Abbildung 1 gibt einen ersten Überblick[4].

Für meine Betrachtungen hier reicht die stark vereinfachte Sicht auf Pattern. Es gibt natürlich mehr Arten von Pattern, aber die drei in der Abbildung 1 beschreiben sehr gut die einzelnen Schritte die notwendig sind, um ein System in einzelne Elemente zu zerlegen.

Ein Architekturmuster beschreibt die Zusammenhang zwischen einzelnen Elemente auf höherer Ebene. Klassisches Beispiel ist dabei das **„Pipes-and-Filter“** Pattern. Hierbei

[1] `https://github.com/tjohann/radio_clock_module`
[2] Gang of Four, siehe [1]
[3] für Link siehe [2]
[4] siehe auch [2]

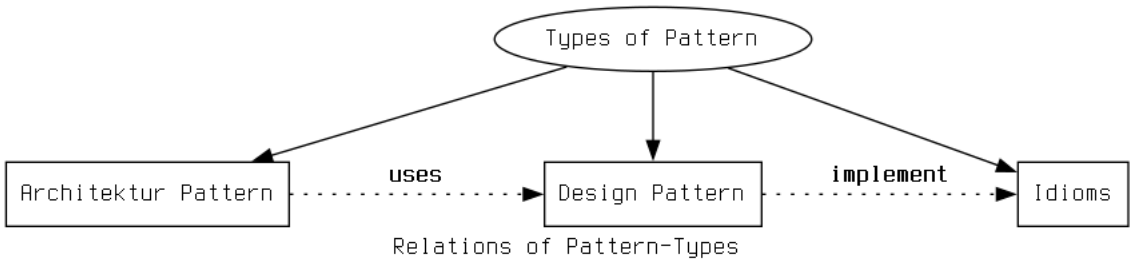

Abbildung 1: Zusammenhang der unterschiedlichen Pattern-Typen

versuche ich einzelne unabhängig aggierende Elemente miteinander über defierte Pfade zu verknüpfen und nutze dabei Filter um die Informationen zu selektieren und zu filtern. Jeder kennt das Beispiel aus der Unix Welt in der einzelen Applikationen über Pipes verknüpft werden können: **„ls | sort | grep | lpr“**. Bei diesem Beispiel sind **sort** und auch **grep** Filter.

Dieses kann ich natürlich auch auf einer noch höheren Ebene einsetzen, z.B. besteht das Beispiel Projekt von mir aus einem Uhr-Modul und einem **externen** Audio-Modul. Da hier die beiden Module über ein Bussystem miteinander verbunden sind[5], würde aber das Patter u.a. aufgrund der Netzwerk-Kommunikation **„Layers“** besser passen.

Weitere interresante Architekturpattern sind **„Model-View-Controller“** und **„Reactor“**. Speziell für mein Beispiel Projekt ist der **„Reactor“** ein möglicher Ansatz um die Trennung des Uhr-Moduls vom eigentlich Steuerungs-Modul, welches ein Raspi-Zero Board ist, zu ermöglichen.

Ein Entwurfsmuster[6] beschreibt jetzt die Architektur einzelner Elemente auf Basis von Standart-Methoden der Realisierung. Als Beispiele kann man die ganzen **Factory Methods** nennen. Die beschäftigen sich u.a. mit der Erstellung von einzelnen Objekten

Die GoF hat die einzelen Designpattern in 3 Gruppen eingeteilt:

- Erzeugungsmuster/Creational Pattern
- Strukturmuser/Structural Pattern
- Verhaltensmuster/Behavioral Pattern

In den weiteren Kapiteln werde ich die Patter beschreiben, die ich in meinem Beispiel Projekt benutzen werde.

Ein Idiom ist die konkrete Realisierung eines Patterns in Code. Aufgrund des Erscheinungsdatums des Buches der GoF sind nicht alle Pattern in dem Buch noch relevant. Diverse von diesen Pattern sind bereits in in C++ eingeflossen, solch eine Realisierung ist ein Idiom. Im Rahmen des Beispiel Projektes werde ich diverse Idiome erzeugen.

[5] ethernet
[6] Designpattern

Literatur

[1] Erich Gamma, Richard Helm, Ralph E. Johnson, John Vlissides; Design Patterns. Elements of Reusable Object-Oriented Software; Prentice Hall; 1st ed., Reprint Edition (1. Juli 1997)

[2] Heise Artikel C++ Blog; `https://www.heise.de/blog/Ein-erster-Ueberblick-Design-Patterns-und-Architekturmuster-mit-C-7147402.html` aufgerufen am 08.01.2023