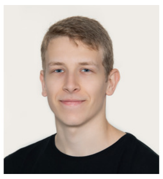
Concept Alternatives for the Management of Architectural Decisions in Clean Architectures

Student



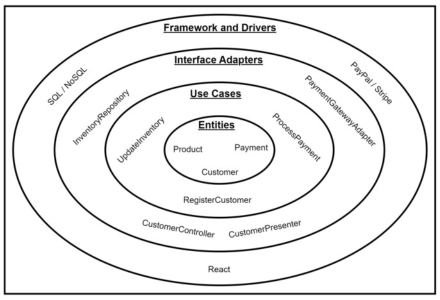
Raphael Schellander

Initial Situation: As software systems become more complex, it is increasingly important to design architectures that can adapt to change, scale effectively, and remain easy to manage. Clean Architecture is a popular approach that helps achieve these goals by organizing systems into layers, each with clear responsibilities. However, applying Clean Architecture can be challenging, especially when it comes to making and keeping track of important architectural decisions. The existing decision management tools often lack domain-specific features and can be difficult to use. There is a need for a better tool that not only helps document decisions, but also guides users through making them in a structured and consistent way.

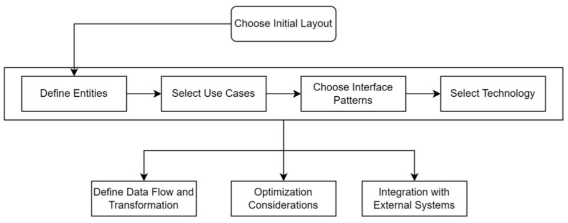
Objective: The objective of this project was to design a new tool concept to address the shortcomings of current tools for managing architectural decisions, particularly in the context of Clean Architecture. This involved analyzing existing tools to understand their limitations, developing a conceptual framework for a new tool that guides users through architectural decision making, and creating an initial proof of concept to demonstrate feasibility. The envisioned tool should help users make decisions in a guided manner and to document those decisions in a way that supports the development of the software. The second objective was to analyze and collect key architectural decisions that repeatedly occur when working with Clean Architecture. The resulting collection is envisioned to guide software architects and developers effectively when they make decisions that align with the key principles of Clean Architecture. This collection also serves as an example of a guidance model that shows the viability of the proposed tool concepts.

Result: The project resulted in an outline for the new tool and a proof-of-concept implementation in Go. The new tool will have an easy-to-use command line interface. It works in combination with the Clean Architecture decision collection that provides concise step-by-step guidance on key architectural decisions such as setting up the initial structure of the system, defining key components and choosing how different parts of the system will interact. The proof-of-concept, while simple, demonstrates how such a tool could work and lays the foundation for further development. Future enhancements could include features such as interactive decision workflows, integration with software development environments, and advanced analysis tools (for example, machine learning) to help make better decisions. Overall, the tool together with the proof-of-concept provide a suited starting point for improving the way in which architectural decisions are made and documented, helping to create more flexible and maintainable software systems.

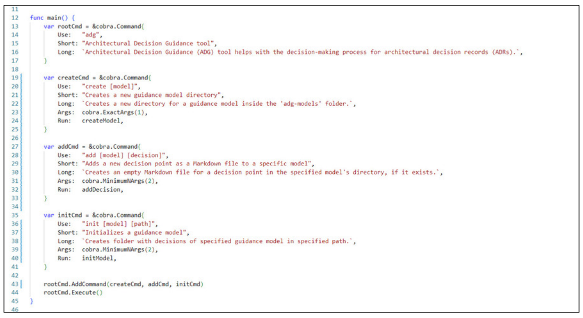
Clean Architecture Layers with Examplary Elements for an E-Commerce Shop
Own presentment



Collection of Key Decisions in Clean Architecture
Own presentment


Choose Initial Layout
Define Entities
Select Use Cases
Choose Interface Patterns
Select Technology
Define Data Flow and Transformation
Optimization Considerations
Integration with External Systems

Proof-of-Concept Source Code (Go)
Own presentment



```go
func main() {
    var rootCmd = &cobra.Command{
        Use:   "adg",
        Short: "Architectural Decision Guidance tool",
        Long:  `Architectural Decision Guidance (ADG) tool helps with the decision-making process for architectural decision records (ADRs).`,
    }

    var createCmd = &cobra.Command{
        Use:   "create [model]",
        Short: "Creates a new guidance model directory",
        Long:  `Creates a new directory for a guidance model inside the 'adg-models' folder.`,
        Args:  cobra.ExactArgs(1),
        Run:   createModel,
    }

    var addCmd = &cobra.Command{
        Use:   "add [model] [decision]",
        Short: "Adds a new decision point as a Markdown file to a specific model",
        Long:  `Creates an empty Markdown file for a decision point in the specified model's directory, if it exists.`,
        Args:  cobra.MinimumNArgs(2),
        Run:   addDecision,
    }

    var initCmd = &cobra.Command{
        Use:   "init [model] [path]",
        Short: "Initializes a guidance model",
        Long:  `Creates folder with decisions of specified guidance model in specified path.`,
        Args:  cobra.MinimumNArgs(2),
        Run:   initModel,
    }

    rootCmd.AddCommand(createCmd, addCmd, initCmd)
    rootCmd.Execute()
}
```

Advisor
Prof. Dr. Olaf Zimmermann

Subject Area
Computer Science