ANASTASIA LABS

Design Patterns - Final Milestone

Project Close-out Report

Project Number	1000012
Project manager	Jonathan Rodriguez
Date Started	2023, October
Date Completed	2024, August

ANASTASIA LABS

Contents

Design Patterns

We would like to briefly take a look back and go over the implemented design patterns

Project Name: Streamlining Development: A User-Friendly Smart Contract Library for Plutarch and Aiken Design Patterns & Efficiency

URL: Project Catalyst Proposal

Introduction

This project aimed to create user-friendly smart contract libraries for Plutarch and Aiken, addressing the challenge of unintuitive design patterns in Cardano development. Our goal was to abstract away complex patterns, making them accessible to developers across the ecosystem while maintaining code readability and efficiency

As developers ourselves, we understood the frustration of dealing with complex patterns that hindered productivity and made it difficult for new developers to enter the ecosystem. We believe that by abstracting away these complexities, we could make Cardano development more accessible to developers across the ecosystem, without sacrificing code readability or efficiency.

Objectives and Challenges

Objectives

- Create comprehensive libraries for both Plutarch and Aiken
- Document key design patterns and efficiency tricks, help developers avoid common pitfalls and optimize their code
- Develop wrapper functions to improve efficiency without sacrificing readability, in general the aim is to make developers' lives easier
- Engage with the community to share knowledge and gather feedback

Challenges Addressed

- Redundant efforts across projects, we saw that many developers were struggling with the same issues and reinventing the wheel
- Complex design patterns and lack of standardization made it difficult to write secure and efficient code increasing the risk of vulnerabilities
- Higher barriers due to a steep learning curve and lack of user friendly resources for new developers entering the ecosystem

Planning

Our project was divided into three main phases:

Design and Documentation

We started by identifying the most unintuitive design patterns and documenting them in detail. We published this documentation on our GitHub repository and contributed to other resources as well

Library Development

We focused on creating reusable, efficient code and implemented key design patterns and wrapper functions for both Plutarch and Aiken, focusing on usability and performance optimization.

Testing

No implementation is complete without thorough testing. We developed comprehensive testing suites, including unit tests and property-based tests, to ensure the reliability and correctness of what we implemented

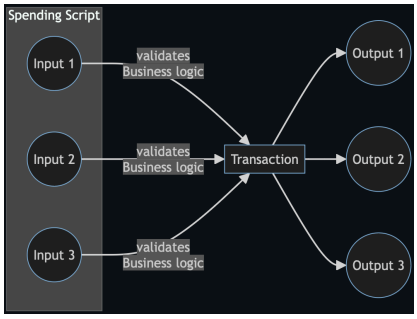
Recap - Design Patterns

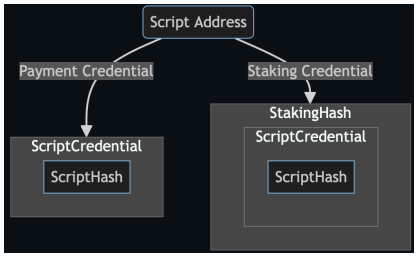
1 - **Enhanced Enum Data Mapping Functions**

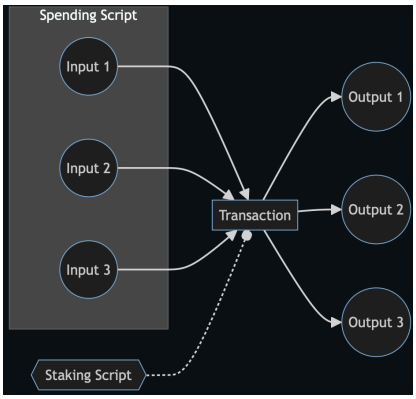
Streamlined implementation of simple redeemers, reducing complexity and lowering costs. This pattern directly maps enumeration cases to integer values, improving efficiency over standard mapping functions

2 - **Stake Validator**

Optimized transaction-level validation using the “withdraw zero trick.” This approach significantly reduced script size and ExUnits cost, with theoretical 5-10x efficiency improvement for transaction-level validation compared to traditional implementations





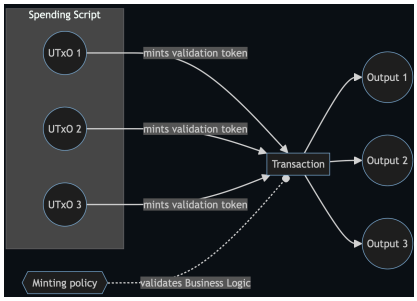


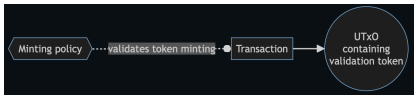
3 - **Merkelized Validators**

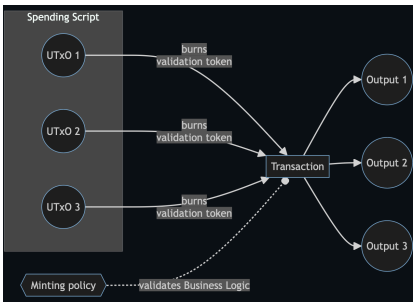
Addressed script size limitations by leveraging reference scripts and the “withdraw zero trick.” This pattern allows for powerful optimizations while keeping main validator size “within limits”, effectively creating smart contracts with near-infinite size potential

4 - **Transaction Level Validation - Minting Policy**

Optimized batch processing of UTxOs by delegating validation to a minting script executed once per transaction. This significantly improves efficiency for high-throughput applications, potentially lowering transaction costs.





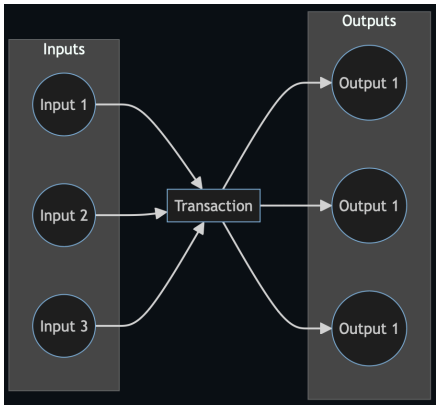


5 - **Strict && Checks**

Addressed inconsistencies in boolean operations across Plutus, Plutarch, and Aiken, providing predictable compilation outcomes and optimizing transaction costs

6 - **UTxO Indexer**

Introduced UTxO indices within the redeemer, allowing validators to efficiently sort and pair inputs with outputs, optimizing transactions with multiple inputs and outputs



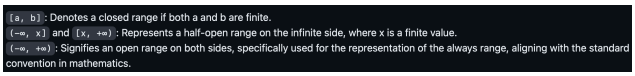
7 - **TxInfoMint Normalization**

Addressed the challenge of automatic 0 Lovelace value appending in txInfoMint field, mitigating unintended consequences

8 - **Validity Range Normalization**

Introduced a normalized representation of time ranges, reducing ambiguity and eliminating redundant or meaningless instances.

```
isTimeValid :: Datum -> POSIXTimeRange -> Bool
isTimeValid datum r =
  case normalizedTimeRange r of
    ClosedRange l u -> ...
    FromNegInf    u -> ...
    ToPosInf    l   -> ...
    Always          -> ...
```



Detailed list of KPIs and references

Challenge KPIs

Performance Optimization

- Optimized mapping functions to reduce complexity and cost of smart contracts
- Managed script size and execution budgets to reduce transaction fees
- Reduced ExUnits cost compared to traditional checks

Security Enhancement

- Measures against known exploits like double satisfaction
- Comprehensive validation by incorporating UTxO indices within the redeemer

Consistency

- Predictable compilation outcomes
- Provided a normalized representation of validity ranges

Project KPIs

Library Completeness

- Inclusion of key design patterns for Plutarch and Aiken

Documentation Quality

- High-quality, detailed documentation for each smart contract library with detailed flow charts/images displaying solution architectures

Engagement

- Active participation in social networks, GitHub, and community events

Key achievements

Development of Comprehensive Libraries
User-friendly libraries for Plutarch and Aiken, simplifying complex design patterns without sacrificing readability and circumventing repetitive boilerplate.

A comprehensive testing suite has been developed utilizing unit and property based tests. More on it can be observed on our extensive **Milestone-4 report**

Exemplary use of these libraries are found for 8 different design pattern scenarios:

- **For Aiken**
- **For Plutarch**

Engagement
This year presentations on our implemented design patterns was given in Buidlfest, a community event specifically scheduled for 100 developers on Cardano. Communication with the developer community is really important to us, as we create tools specifically to make development on Cardano easier day by day.

Examples of the feedback we received during our presentations/on-stage (Toulouse, Buidlfest)

Measurable Result Examples

Transaction-Level Validation Efficiency

Utilizing the “Withdraw” redeemer in staking scripts to run global logic once, rather than for each input. A really significant efficiency improvement over current design patterns can be observed, depending on the number of script inputs in the transaction.

Script Size Reduction

Implementation of techniques like Merkelized Validators to address script size limitations. Potential for “near-infinite” script size while maintaining efficiency, enabling more complex on-chain logic.

Transaction Cost Reduction

Implementation of the UTxO Indexer Design Pattern for efficient sorting and pairing of inputs and outputs. A performance boost for transactions with multiple inputs and outputs. Or the introduction of Stict && Checks for more predictable compilation outcomes and optimized transaction costs

Key learnings

User Feedback

Incorporated feedback from developers/users to improve the libraries

Process Improvements

Development process has been improved based on insights gained during the project development

Best Practices

Documenting best practices for smart contract development and future maintainability and the importance of clear documentation and examples in promoting adoption of advanced design patterns

Next steps

Feature Enhancements

We will maintain and further optimize our existing libraries created for the developers.

Additional design pattern libraries that streamline the implementation process for other existing smart contract languages might come to life as the needs of our developer community requires it. (Such as Scalus, Helios, Plu-ts ...)

Expansion

Targeting a wider developer audience through increased outreach. We are utilizing our design patterns in other tools we develop on Cardano too.

For example, for Lucid Evolution we want to display design patterns in our tutorial series via the evolution library. We strive to create value by making our tools complimentary to each other

Final thoughts

The project successfully addresses its purpose by creating a freely accessible library of design patterns for Cardano developers. Initiatives alike help best practices and already solved puzzles of development on Cardano spread and create ecosystem-wide returns.

We would like to believe our long-lasting open-source efforts have simplified design decisions and improved developer accessibility.

Resources

Project

- GitHub Repository
- Catalyst Proposal

Aiken

- Aiken - Design Patterns
- Test Results / GIF

Plutarch

- Plutarch - Design Patterns
- Test Results / GIF

Close-out Video

- Youtube