Remo
Pfister

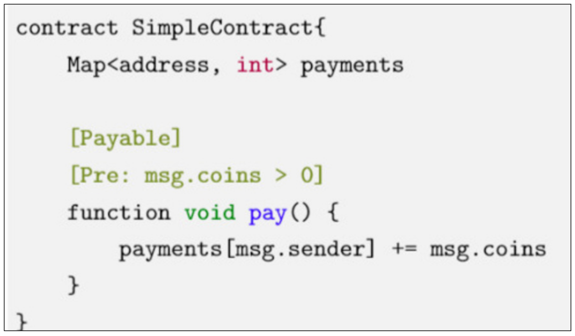
Keerthikan
Thurairatnam

Graduate Candidates	Remo Pfister, Keerthikan Thurairatnam
Examiner	Prof. Dr. Thomas Bocek
Co-Examiner	Sven Marc Stucki, Procivis AG, Zürich, ZH
Subject Area	Software Engineering - Core Systems

Lazo Implementation

Smart Contract Language for Bazo Blockchain

Bazo Logo
https://github.com/bazo-blockchain



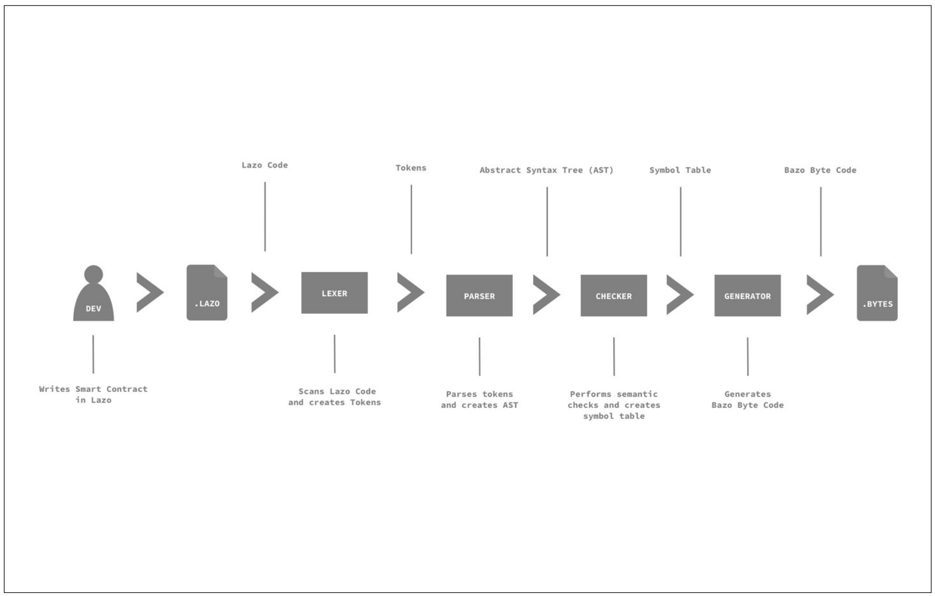
```
contract SimpleContract{
    Map<address, int> payments

    [Payable]
    [Pre: msg.coins > 0]
    function void pay() {
        payments[msg.sender] += msg.coins
    }
}
```

Example Contract in Lazo
Own presentment

Introduction: This bachelor thesis focuses on implementing a compiler for the Lazo programming language. Lazo language is designed to write smart contracts for the Bazo blockchain and was specified during our previous term project. The Bazo blockchain is a research blockchain to experiment with different mechanisms and algorithms.

Approach: Before implementing the language, the language specification and the virtual machine documentation are thoroughly analyzed. Then, the compiler was implemented with the Minimum Viable Product (MVP) approach. Initially, core features were implemented, which are crucial to judge the feasibility of the Lazo language and the capability of Bazo VM to run the Lazo program. After that, more language features were added.
It was decided to use an agile approach to build the compiler. Hence, the features to implement were decided and prioritized over the course of the project in consultation with the supervisor.

Result: As a result, the four-phase compiler, consisting of a lexer, parser, type checker and code generator, was created with all the core features and some additional features such as array, struct, map etc. To verify that the compiler and virtual machine are both compatible, the virtual machine was used to test the generated code and it was possible to find and resolve or report critical bugs in the existing Bazo virtual machine and miner.
The virtual machine was extended with new functionality (opcodes) and existing opcodes were also modified and optimized. Several quality measures were introduced, such as linting, static code analysis, unit and integration tests, code reviews etc.


Lazo Code
Tokens
Abstract Syntax Tree (AST)
Symbol Table
Bazo Byte Code
DEV
.LAZO
LEXER
PARSER
CHECKER
GENERATOR
.BYTES
Writes Smart Contract in Lazo
Scans Lazo Code and creates Tokens
Parses tokens and creates AST
Performs semantic checks and creates symbol table
Generates Bazo Byte Code
Compiler Architecture
Own presentment