Table of Contents

Project overview

Goal

Implemention of smart contract functionality on bazo blockchain. Given use-cases are for examle tokenization, automated refund, identity solution.

Activities

- Research (decision build interpreter or adapt Ethereum/Other VM, figure out, what changes have to be made to the bazo blockchain, find suitable language for smart contracts, define tactic how to implement the solution)
- Implement (build/adapt interpreter/VM, integrate interpreter/vm on bazo, if needed change bazo blockchain)

Scope of supply

- **Project plan:** General overview over the project planning
- **Source code**
- **Github wiki:** Contains pages about research, architecture decisions, set-up manual, user manual, technical documentation)
- **Outline:** Shows the general structure of the thesis, handed in at mid-stage
- **Draft:** handed in after two-thirds of the project is complete
- **Thesis**: Final documentation

- **Personal reports**
- **Presentation**

Management

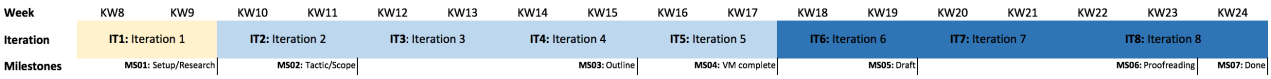
Software development process

We have opted for a fully agile approach since the path of the project is relativly hard to predict. We use a kanbard board and have a backlog to manage the tracking of our tasks and issues.

Shared working hours

We work together on Monday afternoon, wednesday and thursday afternoon.

Time management



Time tracking

In order to track time spent on issues an excel spreadsheet is used.

Milestones

MS01: Project setup and research complete

The dev-team is familiar with Golang, smart contracts, blockchain technology and ready to define the tactic.

MS02: Tactic and project scope defined (Vorgehensweise)

Dev-team is ready to start programming and has laid out a tactic to integrate smart contracts functionality on the bazo blockchain. Also decided between buy vs. build of project components.

MS03: Thesis outline defined

The outline showing the basic structure of the thesis is defined.

MS04: VM complete

The VM is complete and can execute every necessary operation needed for the creation and exection of smart contracts. When this milestone is achieved, the VM can be integrated into the miner.

MS05: Thesis draft defined

Draft showing already more precise structure of the thesis with some chapters completed, some yet to write (i.e. abstract, management summary)

MS06: Thesis ready for proof-reading

One week before the project deadline.

MS07: Project complete

Thesis printed and handed in, code is cleaned up and the documentation is complete. Code is complete if the VM is completly done and integrated into the miner and sample smart contracts can be executed.

MS08: Presentation ready

According to date which is to be defined between 6.-24.08.2018. Done outside the normal project time span.

Meetings

Weekly meetings with advisor on thursday during lunchtime (12:30 or as agreed upon). Protocol is kept in Github wiki.

Risk management

In agreement with the supervisor, the risks were not actively managed.

Infrastructure

Github

Github is used for VCS, Github Projects to manage tasks and the Github Wiki to document the project.

Travis

Travis is used for continuous integration/delivery.

Gitter

Gitter is used to communicate between the dev-team and advisor.

Quality measures

Reviews

With each pull request the code written has to be reviewed by the other team member.

Code quality

Code style

To keep the code well formated, the dev-team has defined gofmt file watchers in GoLand. Every time the file gets saved, the file watcher is executed and the code gets formated.

Unit-Testing

Whenever possible, test-driven development is applied. High test coverage is important.

Versioning

To ensure that the version of any work and code that leaves the VCS can be traced, a version number is defined for each document. Semantic Versioning 2.0.0 is used as a guideline.

Summary Semantic Versioning 2.0.0

The version number is composed of: MAJOR.MINOR.PATCH-LABEL.

- MAJOR version changes with incompatible API changes.
- MINOR version changes with additional functionality which is backward compatible.
- PATCH version changes with backward compatible bug fixes
- LABEL can be used for pre-release or metadata e.g. MAJOR.MINOR.PATCH-alpha.