WingRiders: a decentralized exchange on top of Cardano eUTxO model

WingRiders team

Abstract

In the cryptocurrency space - two key schools have formed when it comes to Centralized versus Decentralized exchanges (DEX). Centralized exchanges (CEM/CEX) are dominant in terms of volume, while decentralized exchanges are superior from a crypto-purist or developer perspective. Decentralized exchanges based on automated market makers (AMM) have become the standard and the most convenient way to exchange crypto tokens. The majority of these DEX's were built on top of account-based blockchains like Ethereum. Cardano's eUTxO model brings exciting benefits that address some of the shortcomings of DEXes on other blockchains with consistent fees and formally verified deterministic behavior. Having the novel eUTxO architecture brings also a fair number of challenges from high-level concurrency issues to low-level technical transaction size limitations. In this paper we describe both the challenges and an initial model for a potential DEX.

Keywords: DEX, Cardano, WingRiders, Vacuumlabs

1. Introduction

We want to provide the community with a trusted, simple and cheap way to exchange cryptocurrency tokens - WingRiders. It is a decentralized exchange (DEX) built on Cardano network. We believe the future of digital money and prosperity exists without an underlying organisation with its hands on the scale.

Automated market makers (AMM) [1] and especially constant function market makers (CFMM) of which the flagship is Uniswap [2, 3] have become a widespread option to implement decentralized exchanges. DEX's typically rely on smart contracts on a permissionless blockchain instead of a (or several) central trusted institutions.

DEX's based on AMMs rely on participants to provide liquidity into liquidity pools (LP) - token pairs locked in a smart contract to facilitate financial transactions. As an incentive to liquidity providers, part of the exchange fees is distributed among them. Part of the fee is in some situations withheld as a protocol fee for parties that are enabling the ability to interact with the distributed financial system. DEX's are controlled by a governance model by the trustless parties. In brief, the code and the smart contract control the interactions between participants - not a third party entity.

The most successful DEX's currently are built with Solidity on Ethereum. Thanks to Ethereum's balance based model and the relative simplicity of AMMs allowed for considerable savings in terms of transaction fees compared to more traditional exchanges based on order books. On the other hand, the non-deterministic fee structure is causing unforeseen complications with collusion and high prices. Think of the recent GAS Fees that have become hotly debated amongst the Ethereum community.

Cardano is a proof-of-stake public blockchain with a transaction based model. The recently launched innovative smart contract architecture on Cardano is radically different from Ethereum's. The architecture is based on the eUTxO [4] model. It is somewhat analogous to bitcoins pay-to-script hash functionality , where the transaction is validated by a script. In Section 2 we give a brief overview of the model and discuss some of its properties in Section 3.

2. eUTxO and Plutus

Cardano's innovative approach to smart contracts is based on the eUTxO model [4]. In this section, we briefly describe the model from the smart contracts viewpoint. For full mathematical description and specification, please refer to Chakravarty et al. [4] and the official specification [5].

The eUTxO model is built upon the basic UTxO (unspent transaction output) model. Each UTxO represents a coin linked to a previous transaction and holds a certain amount of value (native currency). Money is tracked via a chain of custody. UTxOs have to be spent - used as input in a transaction - discretely, but they can be combined or split up as part of transactions. UTxOs in Cardano also contain an address. When spent, an associated signature needs to be provided as a witness in the transaction.

In Cardano's version of the eUTxO model a transaction output is represented as a triplet $(address, value, datum)$. The *value* holds native tokens and currency. The *datum* is an arbitrary JSON-like data attached to the UTxO. The address can either be a public key address or a script address. When a script-address UTxO is spent, the associated validator script is executed to validate the transaction. The script receives the following parameters:

1. Datum of the UTxO,
2. Redeemer with arbitrary data,
3. Context which includes transaction data[1].

A transaction is only valid if all validation scripts for its inputs pass. The redeemer can be viewed as an *intent* how the UTxO will be used by the transaction. The validation scripts are expressed in Plutus code (a Turing complete language). Since all scripts, redeemers, inputs and outputs are known at the time of signing the transaction, the minimum fee required can be deterministically calculated even before submitting a transaction.

As an example for an UTxO validated by a script, lets suppose a vesting "*contract*" which would allow the owner to lock funds in an UTxO for a beneficiary until a given date. The funds could either be withdrawn by the owner before, or by the beneficiary after the deadline. The *datum* in the UTxO could contain the *deadline*, *owner* and $beneficiary$ [2]. The *redeemer* could be either "Withdraw" used by the owner before the deadline or "Vest" used by the beneficiary to transfer out the locked funds. Based on the redeemer, the script validates if the transaction is before or after the *deadline*, and whether the transaction is signed by the *owner* or in case of "Vest" redeemer by the $beneficiary$.

Figure 1 shows two more specifics that are potentially relevant to smart-contract-based DEX's: collateral inputs and minting policy scripts. Transactions on the blockchain are validated in 2 phases: transaction invariants validation and scripts validation. Validating invariants (e.g. input/outputs, hashes, signatures) is relatively cheap and blockchain nodes can reject invalid transactions without any fees. If the validation fails in the second phase, the blockchain nodes need to be reimbursed for the computation time already spent running the validation scripts. For this purpose, *collateral* ADA-only

[1]The context does not include metadata.

[2]For simplicity's sake we are ignoring script parametrization, that would for example allow creating owner and beneficiary specific script addresses. The owner and beneficiary in the Datum would be redundant in this case.

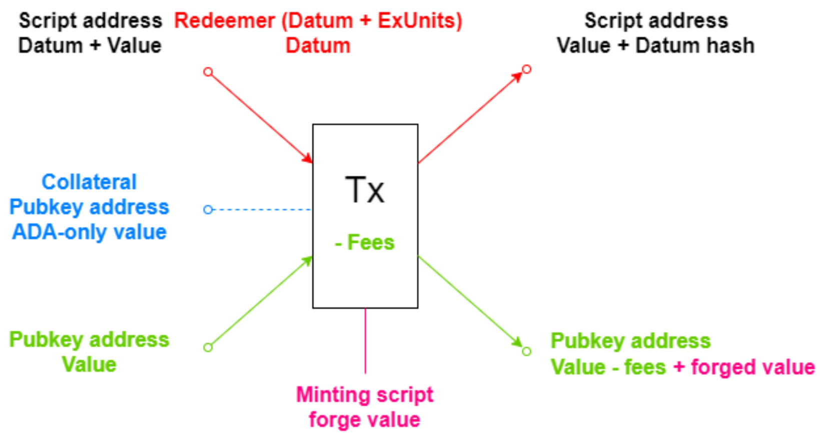

Figure 1: eUTxO transaction model

UTxO(s) need to be provided that are spent entirely (the whole amount). If the validation succeeds, the collateral UTxOs are not spent.

Minting policy scripts, similar to validation scripts are also Plutus scripts. They are used to validate minting of tokens associated with the minting policy. The native tokens forged by the minting policy consequently are identified by $(mintingPolicyHash, tokenName)$.

Smart Contracts

The functionality of traditional smart contracts in Cardano's eUTxO model are covered by two distinct parts:

- **On-chain**: A set of Plutus scripts ran by nodes during the transaction validation. This part ensures the guarantees offered by traditional contracts.

- **Off-chain**: Business logic that builds transactions according to the rules defined in a contract.

The two parts are not necessarily tied together and they don't form a single unit that would be called a "smart contract". Contracts are not *deployed*. Instead, the validator scripts of UTxOs created by the *Off-chain* part define the contract constraints. The UTxOs could be spent by any other transaction not built by the Off-Chain part as long as it adheres to the rules defined in the validation script.

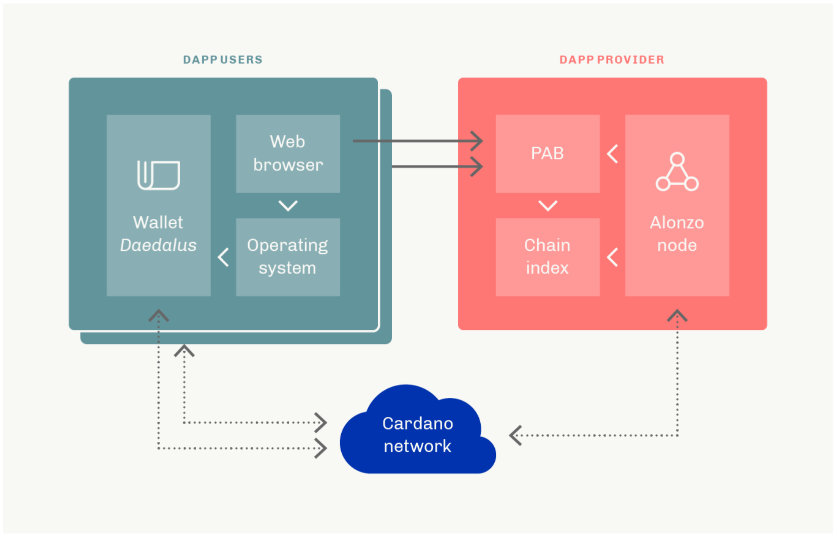

Figure 2: Off-chain logic

The off-chain part as proposed by the Plutus team [6] requires several components demonstrated in Figure 2. The Plutus Application Backend (PAB) would implement the business logic of contracts and would expose an interface to interact with wallets to sign transactions. The off-chain part is currently under heavy development by the Plutus team. We will discuss Plutus Contracts in future iterations of this whitepaper.

The model from Figure 2 is only one of the potential approaches with a dedicated backend PAB service. Given how the on-chain scripts are independent from the off-chain business logic, there is no hard requirement how the transactions spending or creating UTxOs with script validators should be constructed. The business logic could even be implemented as a part of software wallets or tightly coupled with wallets on the front-end.

3. Technical considerations

Ethereum smart contracts allow for great flexibility at the cost of reduced determinism. For example, the exact fee is not known on Ethereum before calling a smart contract, but in Cardano, it can be deterministically calculated *before* submitting a transaction. On the other hand, the eUTxO model also brings unique limitations and challenges.

3.1. Concurrency

Each UTxO can be consumed only once. This causes a concurrency issue: if some funds were locked by a smart-contract only a single actor can interact with it inside one block[3]. On the Ergo blockchain [7] the system uses *read-only* inputs that are not spent by the transaction to solve some of the concurrency issues. For example having read-only inputs from oracles allows multiple smart contract transactions to use the latest oracle value.

In Uniswap, actors interact with the liquidity pool (LP) atomically swapping tokens and updating the LP balances. If the LP was represented as a single UTxO in Cardano, only a single swap could be executed per block. Any other attempts would fail. After every swap a new LP UTxO has to be created with the updated balances. Adding read-only inputs would not solve this concurrency issue.

Strategies to tackle concurrency in Cardano without read-only inputs include: brute-force and bulk transactions. In case of brute-force the smart contract would create x copies of an UTxO. The spender could create transactions against all x copies along with his own UTxO. Only one of these transactions would pass the first phase validation, since the spenders own UTxO can only be spent once. This method would scale the interaction with a smart contract by a factor of x. This approach, however, has limited scalability and usage scope. As transactions increased across the platform, the concurrency issue would become an extreme bottleneck.

“Bulk transactions” strategy splits the interaction with smart contracts into two phases. In the first phase actors with intention to interact with a smart contract would create *request* UTxOs. In the second phase the requests are collected and resolved in bulk in a single transaction. The drawback of this strategy is the trust placed on the party creating the second phase transaction. A big advantage is intuitive user experience as there is no upper limit on the number of requests created in one block, effectively solving the concurrency issue.

3.2. Transaction size

To reduce the UTxO size in Cardano, the UTxOs do not contain the actual Datum values or Scripts. The script is identified by its address (computed from the script hash) and the UTxO only contains the Datum’s hash.

[3]There is a possibility of chaining transactions inside a wallet, but there are no guarantees they would appear in the same block

The script code and datum value need to be provided only as part of the witness by the spending transaction for validation. For performance reasons the transaction size in Cardano is limited by the $maxTxSize$ protocol parameter to $16kB$[4]. The Datum values and scripts take up a significant percentage of the transaction size.

The combination of the above properties leads to a risk of creating unspendable UTxOs if the datum value associated with the hash is larger than the transaction size. Similar logic applies to scripts as well. Sending funds and datum to a large script's address can make the UTxO unspendable if the script is too large.

3.3. Script limitations

The transaction size limitations also become relevant for more complex contracts, where a transaction interacts with multiple UTxOs with distinct script validators. The size of the Plutus scripts empirically varies significantly. For example, the Plutus demonstration use case script sizes vary from $3 - 12kB$[5] at the time of writing. If a set of scripts is used by the contract in the same transaction and share code, it is worth considering merging the scripts into a single validator to reduce transaction size and use redeemers instead as conditions. The trade-off is that merging multiple validators would increase the cost of any transaction interacting only with some parts of the validator.

Apart from script sizes, scripts also have limits in regards to execution steps and memory usage ($(mem, steps) \in ExUnits$). The exact $ExUnits$ need to be provided in the transaction witness as part of the redeemer.

3.4. Minimum ADA value

To limit the maximum total size taken up by UTxO entries in the ledger, the Cardano system imposes a requirement for every transaction output [6] [8]. As a consequence for example, even if a smart contract would only intend to lock some tokens and associated data in an UTxO, some minimum ADA value would still need to be included.

[4]`https://github.com/input-output-hk/cardano-node#network-configuration-genesis-and-topology-files` defined in the Shelley genesis file.

[5]`https://github.com/input-output-hk/plutus/tree/master/plutus-use-cases`

[6]controlled $lovelacePerUTxOWord$ protocol parameter

3.5. Collateral

As described in Section 2, transactions spending UTxOs with validation scripts - or other type of scripts - need to provide ADA-only collateral inputs from a pub-key address. These collateral inputs are only consumed if the script validations fail in the node. The whole collateral inputs are spent and no change is produced. As a consequence, creating transactions using smart contracts adds additional complexity to wallets since they need to prepare and provide additional UTxOs; ideally with limited funds [7]. In theory, if scripts are correctly written and transactions are built correctly according to the constraints, meaning they pass the validation locally in the wallet, the scripts should not fail on the node either. It is one of the selling points of Cardano smart contracts.

3.6. Datum values

As described above, as an optimization, UTxOs do not contain the datum value, only the hash. On the other hand, the datum value needs to be included by the spending transaction witness. The PAB, wallet or other service creating the transactions needs access to the original data to be able to spend the UTxO. In Figure 2, for the proposed off-chain solution, it is the *Chain-index* that provides the lookup $DatumHash \rightarrow Datum$. The Chain-index is sourced from the blockchain node, hence can only include lookup for Datums already available on the blockchain. For the initial spender of an UTxO with an unknown hash, a different mechanism needs to be in-place.

4. AMM implementation on Cardano

As described in Section 1, AMM based exchanges became the the most common approach to implement decentralized exchanges. This section describes a potential adaptation of a generic DEX with constant-function market maker for the Cardano blockchain. Individual parts of the model will be discussed in more detail later in this section. For a very brief simplified overview: the model is based on a set of *liquidity pools* (LP). Each LP contains funds for a pair of distinct tokens (A, B). Participants can use LPs to

[7]The minimum total value of the collateral is controlled by the $collateralPercentage$ protocol parameter.

swap their A tokens for tokens B by adding tokens A into the pool and withdrawing tokens B and vice versa. The exchange price is determined based on the ratio of $r_A : r_B$, such that

$$r_A \cdot r_B = k \tag{1}$$

is kept constant (where r_X are the reserves for token X).

Participants can also add funds into the LP in exchange for liquidity pool tokens. For each swap a percentage of the funds - a swap fee - is returned into the LP as an incentive and reward for liquidity providers. Providing initial liquidity into an LP is done through a liquidity pool factory contract. The pool factory tracks all available LPs to ensure their uniqueness.

Even though the model is conceptually simple, there is a wide range of possibilities how it can be implemented for the Cardano blockchain. The guarantees we aim to provide consist of:

1. **Trust**: Funds sent to an address in order to provide liquidity or swap tokens would not be lost. Scripts validating the use of the funds will be audited and open source.
2. **Fairness**: There is no bias towards any particular participant and the swaps are executed in a canonical order.
3. **Reliability**: Any funds locked in scripts can be reclaimed.
4. **Security**: The source code will be audited and critical parts will be open source. In addition, smart contracts in Cardano have smaller scope compared to Ethereum leaving less room for error.
5. **Anonymity**: Anyone with ADA and native tokens can use the DEX. We impose no other requirements. Hence, the DEX is as anonymous as the Cardano blockchain is.
6. **Simplicity**: We aim to provide the simplest and clearest UX for users to understand the system easily.
7. **Affordability**: We aim to minimize transaction fees required to interact with the DEX by minimizing script, datum and value sizes. Additionally, the transaction fees on Cardano are deterministic and significantly smaller than those on Ethereum.
8. **Opportunity**: Apart from liquidity pool rewards, we aim to cover delegating ADA for staking in the design.

4.1. Liquidity pool factory

The liquidity pool factory can be represented as a script UTxO made of:

1. NFT to ensure uniqueness,
2. datum containing the list of already available token pairs,
3. validator script that allows creating new LPs.

When a new LP is created, the pair is added to the list. There is a scalability concern regarding the ability to store large number of token pairs and the risk of creating an unspendable factory UTxO (see Section 3.2).

LPs associated with the DEX can only be created by the factory. To ensure validity of LPs on the blockchain, they have to carry the token minted by the factory's minting policy. The minting policy would need to be part of the transaction where a new LP is created. As highlighted in Section 3.3 there needs to be conscious effort to fit both validator and minting policy scripts along with a potentially large datum into the transaction size limits. The creator of the new LP would also need to provide the minimum ADA value for the new LP UTxO (see Section 3.4).

Solutions to the identified challenges will be discussed in future publications.

4.2. Liquidity pools and swaps

The liquidity pools associated with a token pair can be represented as UTxOs that have:

1. a validity token provided by the liquidity pool factory,
2. reserves for both tokens,
3. datum with supporting information,
4. validation script that allows swaps, adding liquidity and withdrawing liquidity and ensures that an updated LP UTxO is created.

The spending transaction would need to provide the correct redeemer to differentiate between the actions. With each transaction the spender would also be required to provide collateral (see Section 3.5). The largest flaw of such representation, however, is the issue around concurrency described in Section 3.1. With a single LP UTxO only one swap request could be executed per block (roughly every $20s$).

Another potential approach is to use bulk transactions (see Section 3.1). In case of swap operations, the participants could create *Swap Request* UTxOs, which would be executed (=spent) in bulk in a single transaction as shown in Figure 3. This approach, however, raises trust issues. The agent could for

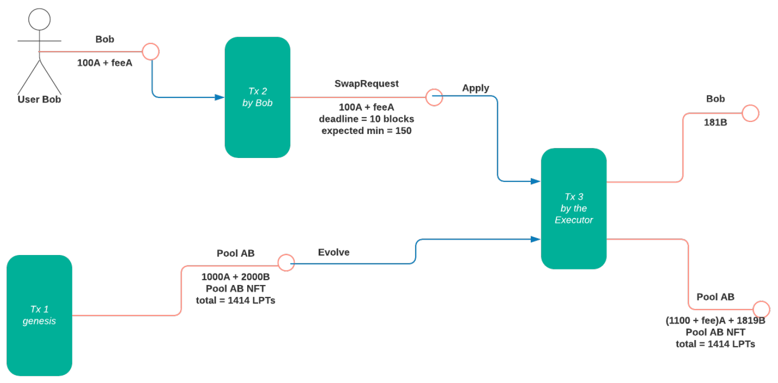

Figure 3: Bulk transaction executing the swap request

example omit a Swap Request or change the order in which they are applied to enable front-running.

Since Cardano is a proof-of-stake blockchain, liquidity providers of $(ADA, token)$ pairs would sacrifice the opportunity to delegate their funds to a stake pool and earn rewards. Another minor inconvenience of the bulk transaction strategy is the minimum ADA requirement for all Swap Request UTxOs as described in Section 3.4.

We will discuss the concurrency, scalability and staking solutions in future publications.

4.3. Fees

As described at the beginning of Section 4, to reward liquidity providers, a percentage of the swapped volume is returned into the LP pools (e.g. 0.3% in Uniswap v2 [2]). The exact fee structure will largely depend on the concurrency strategy. For example, if the bulk transaction strategy was used, part of the fees would need to be allocated towards the executor to cover the bulk transaction fees and collateral. We will discuss fees in full detail in future publications.

5. Governance and Tokenomics

Similar to fees (see Section 4.3), the exact governance will also largely depend on the concurrency strategy deployed. For example, in case of the "bulk transaction strategy", the governance token holders could be able to vote on assigning the *executor token* to a selected pubkey address. Holding the *executor token* would enable creating bulk transactions for swap requests.

As we approach the project with a "Developer mindset" - We plan to cover governance around:

- **Upgrades**: The initially released version probably won't be the most optimal given how quickly the ecosystem is evolving. We plan to iterate on the best solution for the largest possible audience and ease the transition to newer versions in the most secure way possible.
- **Fees**: Controlling the fees and their distribution between the liquidity pool providers and other entities.
- **Staking**: Delegating ADA locked in scripts to staking pools should be governed by the LP token holders.
- **Rewards**: A part of the project tokens will be reserved as rewards for liquidity pool providers for locking their funds in the LPs for an extended period of time.

Tokenomics will be finalized before the official release.

6. Future work

The DEX solution above can be extended in multiple ways. In the future academic papers we plan to focus on the following aspects among many:

1. **Routing**: The DEX design above assumes only direct swaps for the available liquidity pool pairs. Chained swaps against multiple LPs with the above representation cannot be executed atomically on Cardano.
2. **Range pools**: The innovation from Uniswap v3 [3] would offer better economic opportunities for liquidity pool providers.
3. **Price oracles**: We aim to support other smart contracts by providing them verified information and enable secondary derivative markets.

7. Conclusion

The release of smart contracts on to the Cardano blockchain brings exciting new challenges and opportunities. The combination of on-chain and off-chain parts allow for great flexibility in terms of deploying services, while also ensuring very strict rules and validation for transactions. We discussed some of the challenges associated with creating a constant-function market maker distributed exchanges, most notably the concurrency issue. We believe the solution described in this document would lay the foundation for more sophisticated, more secure and scalable designs in the future.

Disclaimer

This paper only serves as an initial conceptual proposal. A detailed DEX architecture that addresses all listed challenges and offers precise economic calculations and comparisons will be made public after the initial release of the DEX.

The ecosystem around Cardano smart contracts is still under heavy development with several key pieces still being finalized. As new research and products see the light, this document is subject to revisions until the initial release.

This paper is for WingRiders information purposes only. It does not constitute investment advice or a recommendation for solicitation to buy or sell any investment and should not be used in the evaluation of the merits of making an investment decision. It should not be relied upon for accounting, legal or tax advice or investment recommendations. This paper reflects current opinions of the authors and is not made on behalf of Vacuumlabs or their affiliates and does not necessarily reflect the opinions of Vacuumlabs, their affiliates or individuals associated with them. The opinions reflected herein are subject to change without it being updated.

References

[1] A. Othman, Automated Market Making: Theory and Practice, Ph.D. thesis, Carnegie Mellon University, 2012.

[2] H. Adams, N. Zinsmeister, D. Robinson, Uniswap v2 Core (2020).

[3] H. Adams, N. Zinsmeister, M. Salem, R. Keefer, D. Robinson, Uuniswap v3 Core (2021).

[4] M. Chakravarty, J. Chapman, K. MacKenzie, O. Melkonian, M. P. Jones, P. Wadler, The Extended UTXO Model, Technical Report, IOHK and University of Edinburgh, 2020.

[5] A. Knispel, P. Vinogradova, A formal specification of the cardano ledger integrating plutus core, https://hydra.iohk.io/job/Cardano/cardano-ledger-specs/specs.alonzo-ledger/latest/download-by-type/doc-pdf/alonzo-changes, 2021.

[6] Alonzo-testnet, Plutus application backend (pab) explainer, https://github.com/input-output-hk/Alonzo-testnet/blob/main/explainers/PAB-explainer.md, 2021.

[7] E. Developers, Ergo: A resilient platform for contractual money (2019).

[8] C. L. Specs, Min-ada-value requirement, https://github.com/input-output-hk/cardano-ledger-specs/blob/master/doc/explanations/min-utxo.rst, 2021.