**Simulation for Ethereum Applications**

**Goals:**

* Have a system of smart contracts +
* Pool of users that want to use said contracts
* See what happens when you let them use it for a given period of time

**System of Smart Contracts**

One or more smart contracts tied together acting as a unit

You get compiled bytecode, runtime bytecode, ABI (JSON encoded string that lists methods and events that the contract has. JSON list, with each element either a function (returns) or an event (doesn’t return)

One ABI and one set of bytecode per smart contract – Gets compiled with solidity compiler (input)

**Pool of Users**

One or more types of users that can be provisioned into x amount of instances of

Basic user class, admin class; eg, 99 basic users and 1 admin

User classes need to execute function calls and listen for events from the ABI of the smart contracts they interact with. They need to be able to have programmable behavior (ie, if it receives x > y tokens then it’ll do z) Can read from ETH to make decisions and listen to events to get callbacks to decisions.

**Run a simulation**

Provision x amount of user classes (instances), which have preprogrammed behaviors.

Deploy the system of smart contracts to blockchain

1. Upload system of smart contracts to ETH blockchain
2. Provision the user agents, which will need to know the addresses as input params
3. Kick off a simulation
   1. Basic simulation – scenarios
   2. Go through time; start with 20 users, and then the trees expand
   3. Basic cycle method: running all the stuff on the low-end
   4. Scenarios class
   5. Optional: kick off several scenarios at once
4. Output: JSON output of the state (per block) for time-based analysis