Smart Contracts

Blockchain Technology (CS467)

B.Tech. IV, Semester VII

Department of Computer Science and Technology
S. V. National Institute of Technology-Surat

Smart Contracts

- Executable code
- Turing Complete
- Function like an external account
  - Hold funds
  - Can interact with other accounts and smart contracts
  - Contain code
- Can be called through transactions

Smart Contract

- Replaces traditional contracts in blockchain context
- Authority-less autonomous program, that directly controls numeric securities (digital assets), based on mutually agreed terms
- Looks like “if-then” instructions that automatically evaluate predefined conditions and do transactions
- Has an owner and a life cycle, and is executed on Ethereum Virtual Machine (EVM)

Smart Contract Advantages

- Immutability characteristic Contract terms will not change
  - How about when a bug is introduced on a smart contract?
- Autonomy and automaticity characteristics
  - Reduce audit and execution cost, and fraud
  - Allows to restrict actions as for acquisition schedule, where an individual owns actions but cannot dispose of them before a given date

Smart Contract Limitations

- Not suitable for all kinds of contracts such as legal ones
- Can only “solve issues which can be objectively decided based upon the facts”
- Simple and only include “if a, then b” patterns
- Legal contracts allow to make a subjective judgement even if no objective facts were transcribed such as “without undue delay” and “beyond a reasonable doubt”

Code Execution

- Every node contains a virtual machine (similar to Java)
  - Called the Ethereum Virtual Machine (EVM)
  - **Compiles** code from high-level language to bytecode
  - Executes smart contract code and broadcasts state
- ***Every full-node on the blockchain processes every transaction and stores the entire state***

Gas

- Halting problem (infinite loop) – reason for Gas
  - Problem: Cannot tell whether or not a program will run infinitely from compiled code
  - Solution: charge fee per computational step to limit infinite loops and stop flawed code from executing
- Every transaction needs to specify an estimate of the amount of gas it will spend
- Essentially a measure of how much one is willing to spend on a transaction, even if buggy

Gas Cost

- Gas Price: current market price of a unit of Gas (in Wei)
  - Check gas price here: https://ethgasstation.info/
  - Is always set before a transaction by user
- Gas Limit: maximum amount of Gas user is willing to spend
- Helps to regulate load on network
- Gas Cost (used when sending transactions) is calculated by gasLimit*gasPrice.
  - All blocks have a Gas Limit (maximum Gas each block can use)

PoW vs. PoS

Ethereum in the process of moving to Proof of Stake

- This approach does not require large expenditures on computing and energy
- Miners are now “validators” and post a deposit in an escrow account
- The more escrow you post, the higher the probability you will be chosen to nominate the next block
- If you nominate a block with invalid transactions, you lose your escrow

PoW vs. PoS

Ethereum in the process of moving to Proof of Stake

- One issue with this approach is that those that have the most ethereum will be able to get even more
- This leads to centralization eventually
- On the other hand, it reduces the chance of a 51% attack and allows for near instant transaction approvals
- The protocol is called Casper and this will be a hard fork

https://blockonomi.com/ethereum-casper/

Other approaches to conensus

There are many other types of consensus

- (PoW) Proof of Work (Bitcoin, Ethereum, …)
- (PoS) Proof of Stake (Ethereum in future)
- (PoI) Proof of Importance (used in NEM)
- (PBFT) Practical Byzantine Fault Tolerance (Hyperledger Fabric)
- (FBFT) Federated Byzantine Fault Tolerance (Ripple, Stellar)
- (DPoS) Delegated Proof of Stake
- (PoET) Proof of Elapsed Time (Hyperledger Sawtooth)

https://medium.com/@chrshmmmr/consensus-in-blockchain-systems-in-short-691fc7d1fefe

B. Smart Contract Programming

- Solidity (javascript based), most popular
  - Not yet as functional as other, more mature, programming languages
- Serpent (python based)
- LLL (lisp based)

B. Smart Contract Programming

Solidity

Solidity is a language similar to JavaScript which allows you to develop contracts and compile to EVM bytecode. It is currently the flagship language of Ethereum and the most popular.

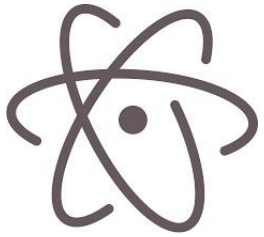
- Solidity Documentation - Solidity is the flagship Ethereum high level language that is used to write contracts.
- Solidity online realtime compiler

Serpent

Serpent is a language similar to Python which can be used to develop contracts and compile to EVM bytecode. It is intended to be maximally clean and simple, combining many of the efficiency benefits of a low-level language with ease-of-use in programming style, and at the same time adding special domain-specific features for contract programming. Serpent is compiled using LLL.

- Serpent on the ethereum wiki
- Serpent EVM compiler

B. Smart Contract Programming



Atom Ethereum interface - Plugin for the Atom editor that features syntax highlighting, compilation and a runtime environment (requires backend node).

Atom Solidity Linter - Plugin for the Atom editor that provides Solidity linting.

Vim Solidity - Plugin for the Vim editor providing syntax highlighting.

Vim Syntastic - Plugin for the Vim editor providing compile checking.

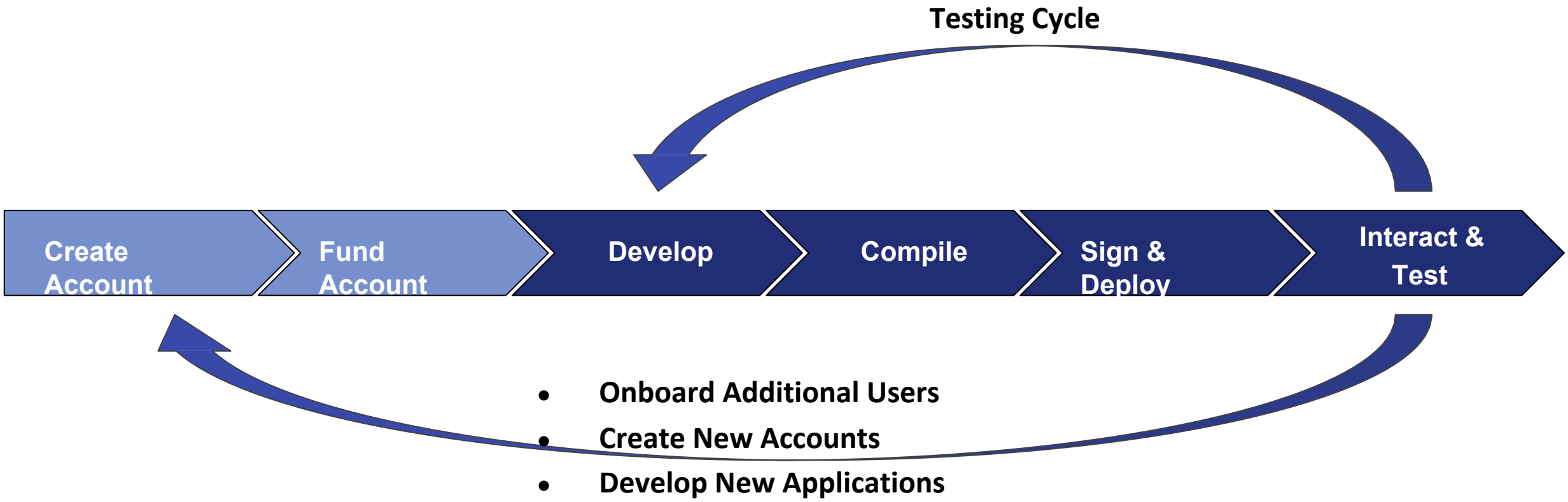
B. Smart Contract Programming: Solidity

```
contract Example {

    uint value;

    function setValue(uint pValue) {
        value = pValue;
    }

    function getValue() returns (uint) {
        return value;
    }

}
```

B. Smart Contract Programming: Solidity

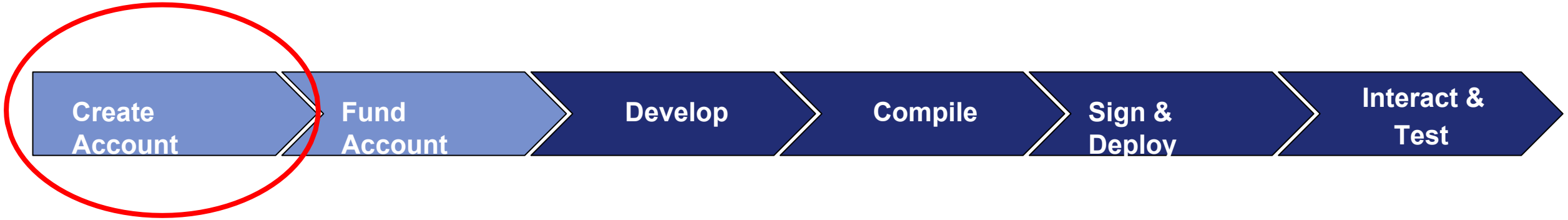
```
var logIncrement =
   OtherExample.LogIncrement({sender: userAddress,
uint value});

logIncrement.watch(function(err, result) {
  // do something with result
})
```

C. Development Workflow



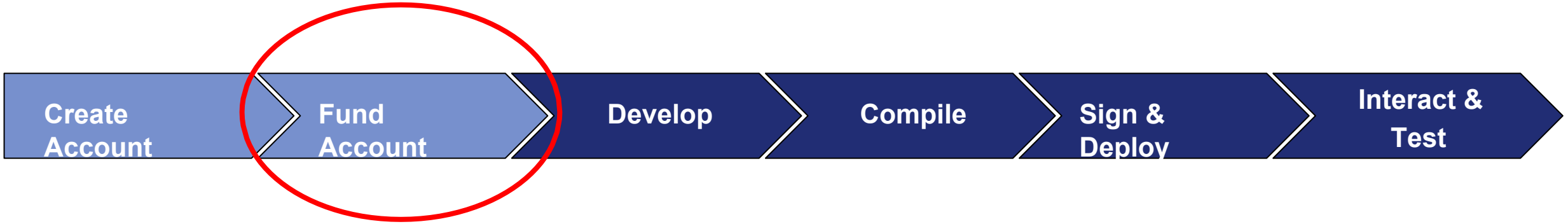
- **Onboard Additional Users**
- **Create New Accounts**
- **Develop New Applications**

C. Development Workflow: Create Account



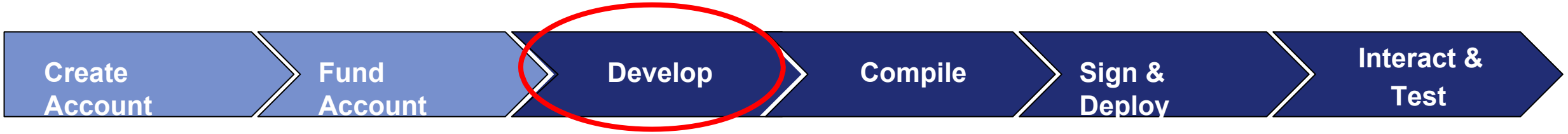
- Programmatically: Go, Python, C++, JavaScript, Haskell
- Tools
  - MyEtherWallet.com
  - MetaMask
  - TestRPC
  - Many other websites

C. Development Workflow: Fund Account



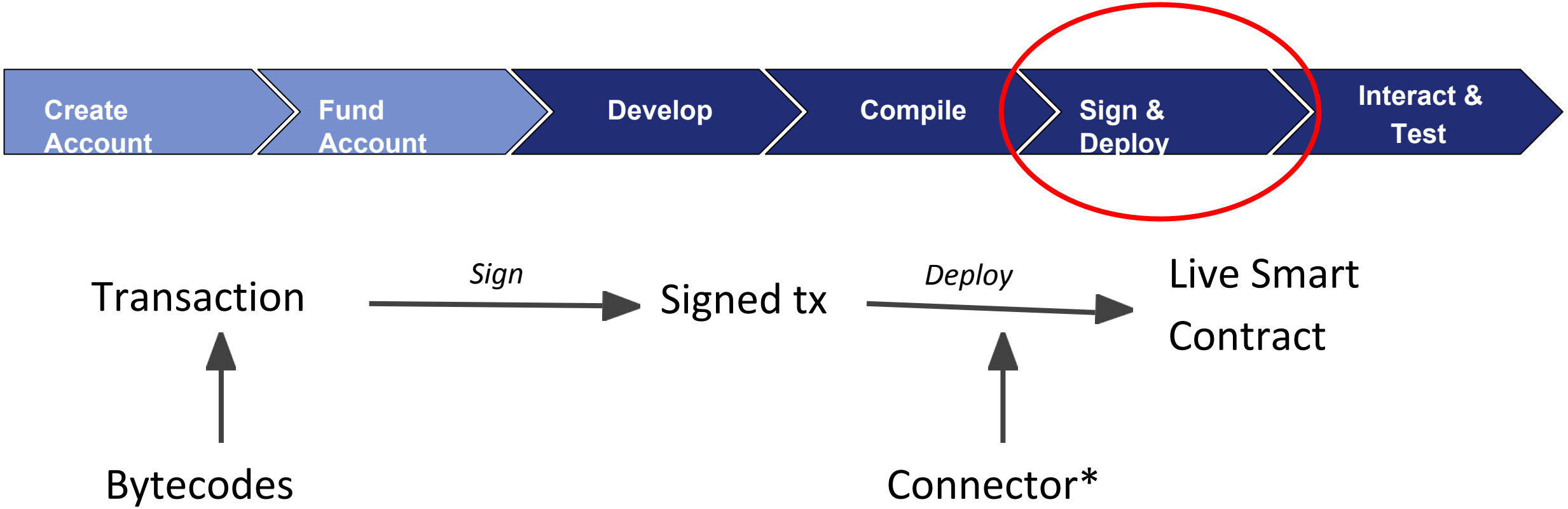
- From friends
- Faucet
- Exchanges (for public blockchain)

C. Development Workflow: Develop



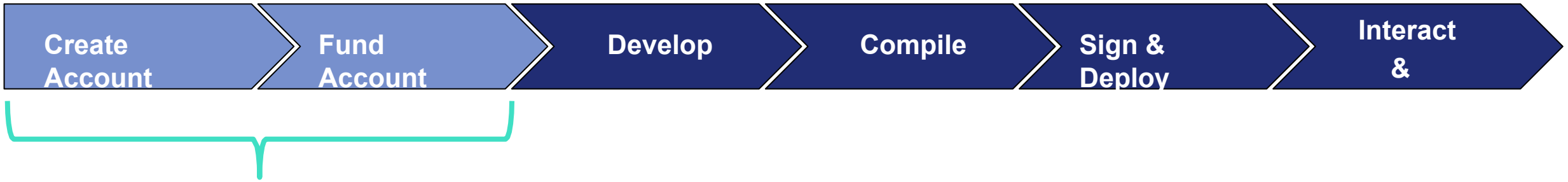
- **Ethereum Application Components:**
  - **Base application**: can be developed in **any** language
  - **Smart contract**: developed in Solidity or one of the other contract compatible languages
  - **Connector library**: facilitates communication between base application and smart contracts (Metamask)

C. Development Workflow: Sign and Deploy



*Library that facilitates communication and connection with Blockchain; Connects your code to a running node.

C. Development Workflow: TestRPC



TestRPC/TestChain

- Local development or Test Blockchain
- https://github.com/ethereumjs/testrpc