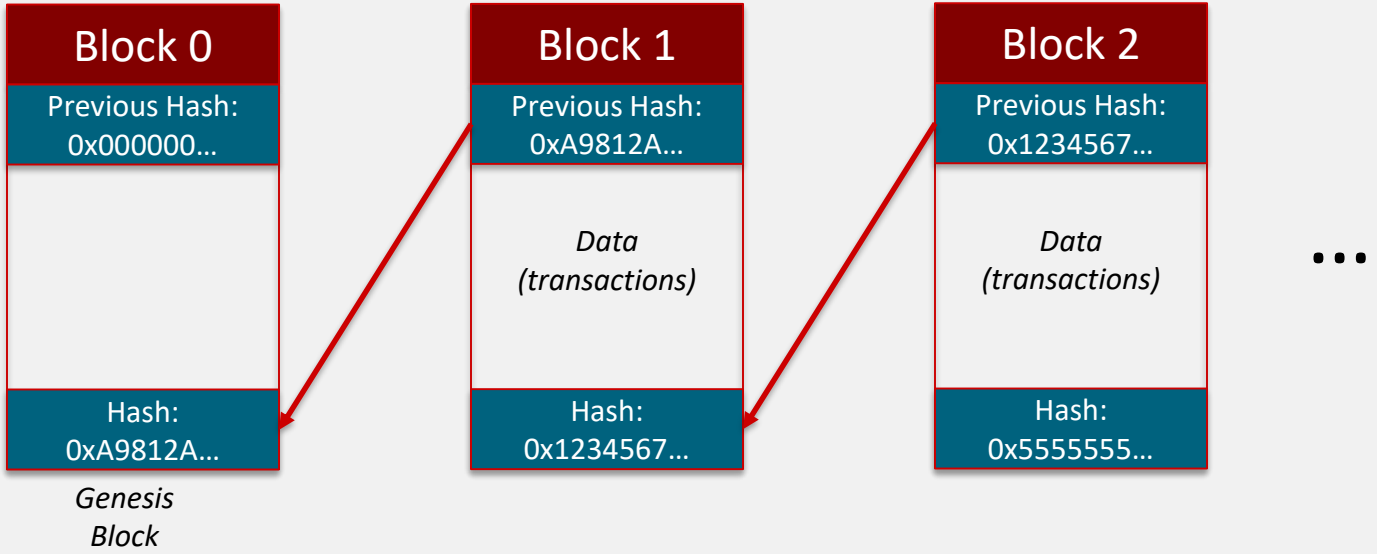
redhat.

SMARTER CONTRACTS WITH DMN

Leveraging DMN to improve Smart Contract based solutions

Edson Tirelli
Principal Software Engineer
Drools Project Lead

BLOCKCHAIN?



BLOCKCHAIN

Characteristics

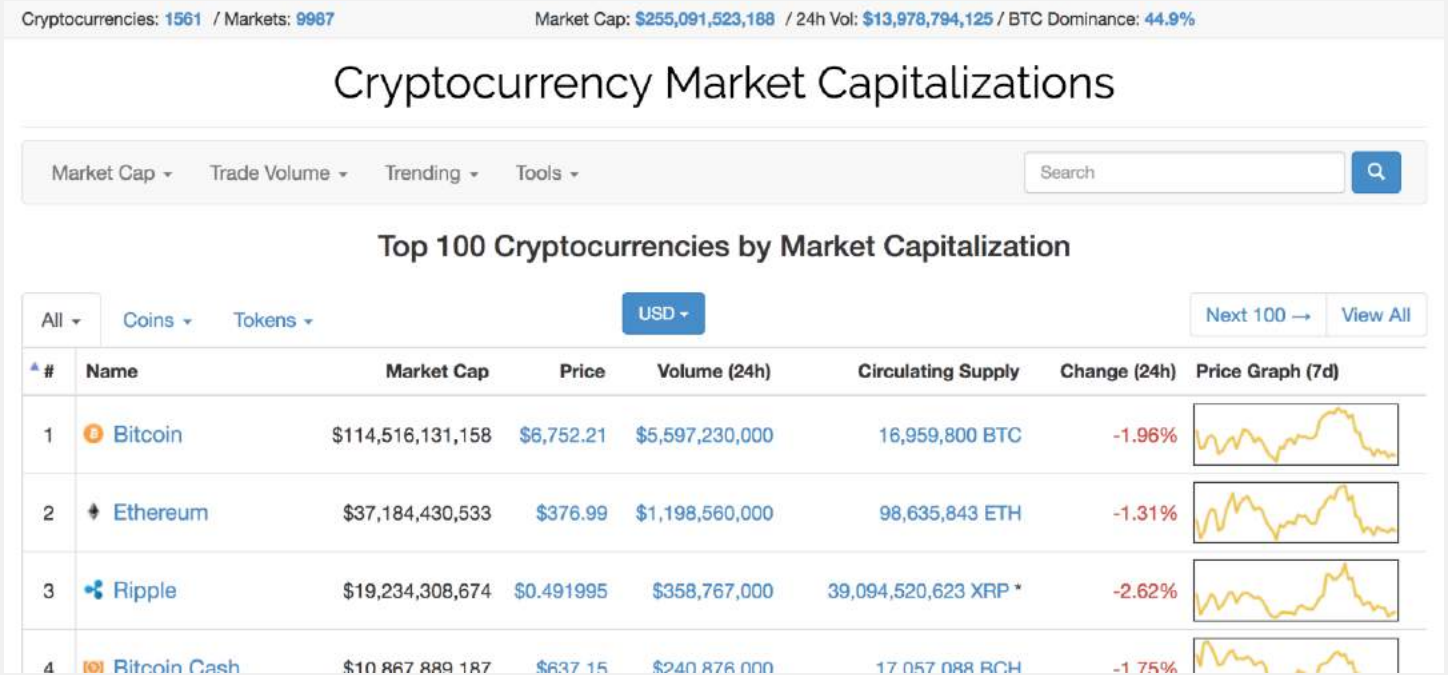
- **Distributed:** replicated on multiple network nodes
- **Decentralized:** no central authority, based on consensus
- **Trustless:** peers don’t need to trust each other, simply trust the protocol
- **Public:** no hidden or private data within the network, although some data can be encrypted
- **Append-only**: no data can be changed without invalidating the subsequent chain of blocks

CRYPTOCURRENCY?

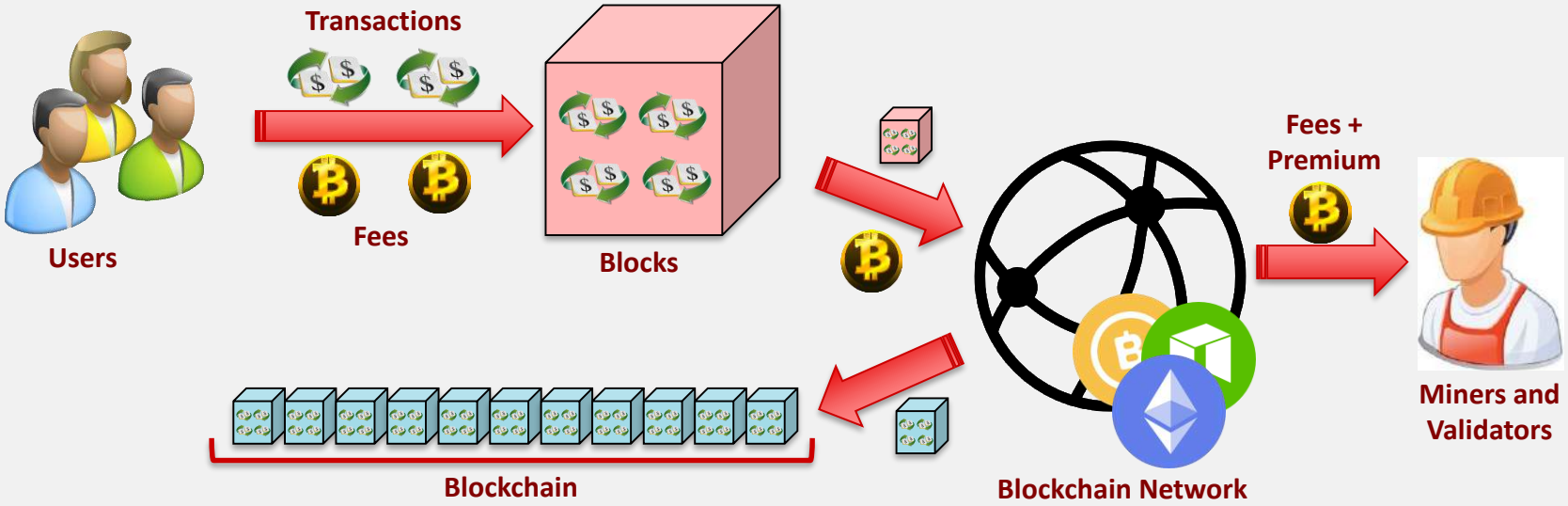
- **Digital Currency:** trade recorded in blockchain ledgers
- **Virtual:** typically no real world asset backing, i.e., no intrinsic value
- **No central authority:** creation and disposal methods predefined, built into the network
- **Helps moderate blockchain operation:** used to pay the costs of processing transactions in the blockchain

BLOCKCHAIN

Implementations

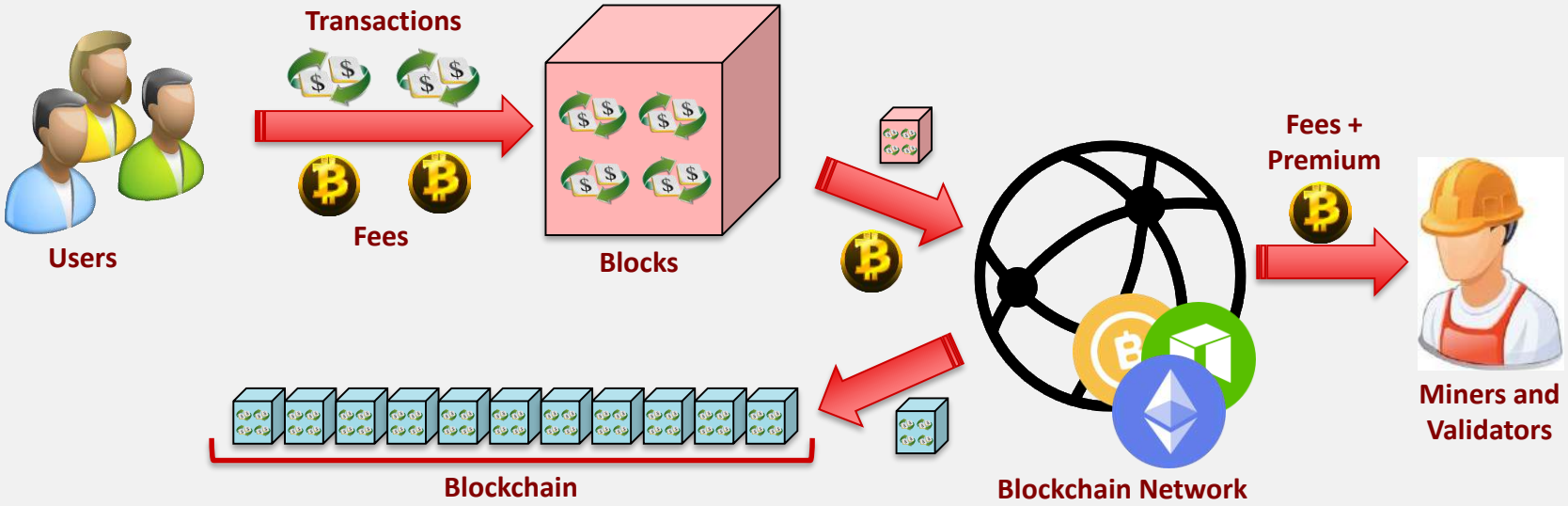


BLOCKCHAIN TRANSACTION WORKFLOW



Source: *https://blockgeeks.com/guides/what-is-blockchain-technology/*

BLOCKCHAIN TRANSACTION WORKFLOW



Source: *https://blockgeeks.com/guides/what-is-blockchain-technology/*

SMART CONTRACTS?

Distributed, decentralized, automated applications that run on top of a blockchain network.

E.g.: a logistics contract that tracks items through fulfillment.

Source: *https://www.quora.com/What-are-smart-contracts*

SMART CONTRACT USE CASES

- Digital Identity management
- Securities
- Multi-party B2B settlement
- Mortgages and Land Title Registry
- Supply chain (including IoT integration)
- Food sourcing
- Insurance
- Clinical trials

ETHEREUM?

A blockchain implementation focused on enabling development of Decentralized Applications (DAPPs).

Implements a distributed, Turing Complete, virtual machine for the execution of code: the Ethereum Virtual Machine: EVM

Also: created its own cryptocurrency, called "Ether"

SMART CONTRACTS IN ETHEREUM

```
1  contract MetaCoin {
2    mapping (address => uint) balances;
3
4    function MetaCoin() {
5      balances[tx.origin] = 10000;
6    }
7
8    function sendCoin(address receiver, uint amount) returns(bool sufficient) {
9      if (balances[msg.sender] < amount) return false;
10     balances[msg.sender] -= amount;
11     balances[receiver] += amount;
12     return true;
13   }
14
15   function getBalance(address addr) returns(uint) {
16     return balances[addr];
17   }
18 }
19
```

Example contract in Solidity: *a "meta-coin" implementation*

Source: *https://medium.com/pactum/what-is-a-smart-contract-10312f4aa7de*

- *Written on a language that compiles to EVM bytecode. Most popular: Solidity, Serpent, LLL and Mutan.*
- *Contracts can be deployed into the network as a regular transaction.*
- *At deployment, a contract is assigned an "address", or account, similar to user accounts (a.k.a. wallets)*
- *Interactions with the contract that change state, are also transactions.*

THE “ECONOMICS” OF SMART CONTRACTS

- All transactions in a blockchain cost money: the transaction fee
- All interactions with a contract that change state are transactions
- Ergo, interacting with a smart contract to change state costs money

Operation	Cost in “*Gas*”
Storing a byte of data	5 gas
Adding two numbers	3 gas
…	…

Cost in *Gas* for each operation is constant

Cost of *Gas* in Ether is variable: e.g. 15e-10 Ether/gas unit

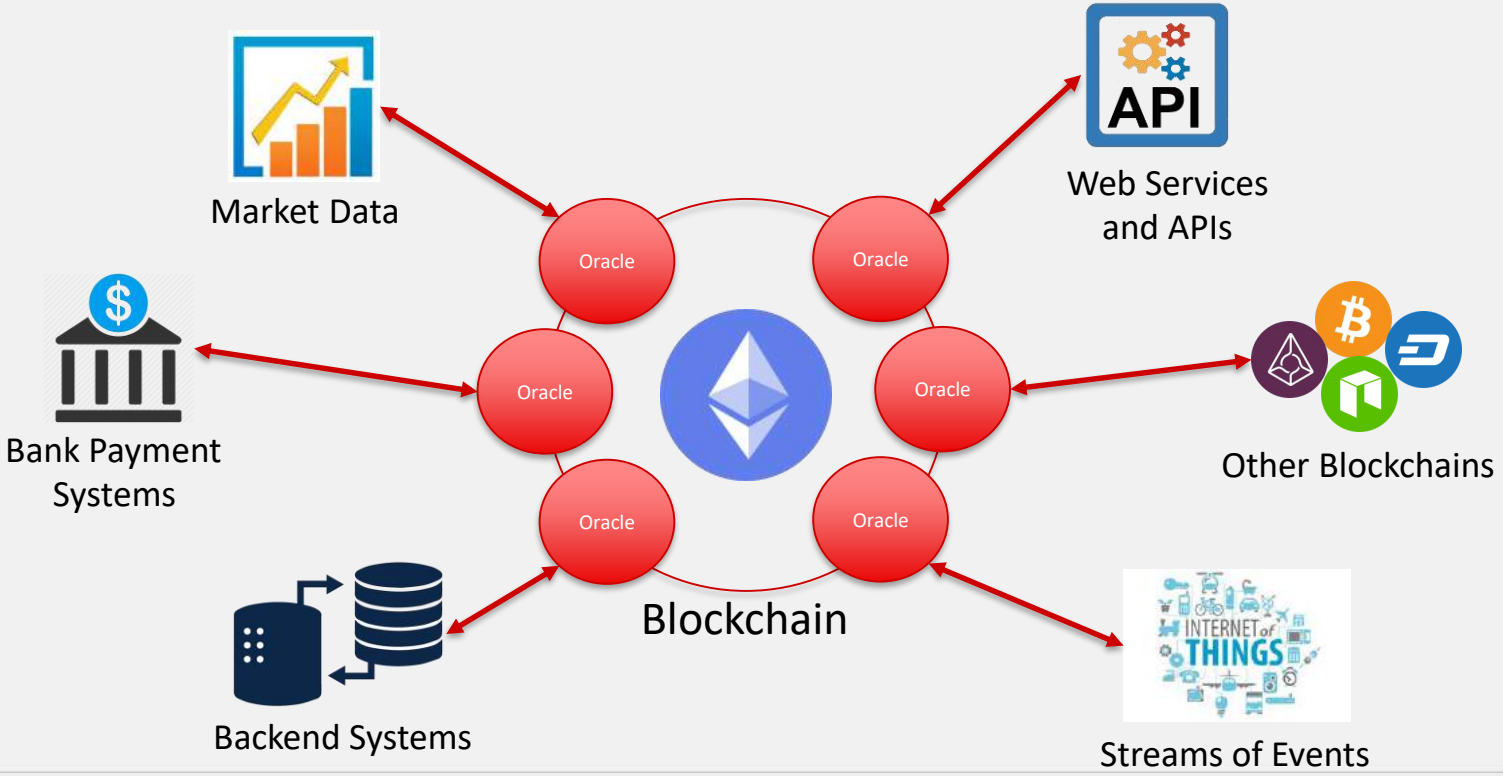
THE ETHEREUM TRANSACTION LIMIT

- Ethereum defines a **variable hard limit** on the gas cost of transactions (as of April 6th, 2018, ~8 million gas units)
- Transaction hits hard limit = transaction is aborted and rolled back
- Helps **prevent attacks or mistakes** that would compromise the stability of the network
- Smart Contracts need to operate within transaction limits:
  - Loops/iteration in contract code = bad
- **Invoker** of the Smart Contract operation **pays the cost**

ON-CHAIN/OFF-CHAIN PROCESSING

- EVM (Ethereum Virtual Machine) is **Turing Complete**: can execute any computation
- EVM is a **sandbox**
  - because code needs to be executable on all nodes of the network
  - protects integrity of the network
- Smart Contracts have **no direct external access**
  - no interface to legacy applications or services
  - no input/output interfaces (in the traditional sense)
- Smart Contracts code is executed by the EVM, i.e., **ON-CHAIN processing**

ORACLE SERVICES

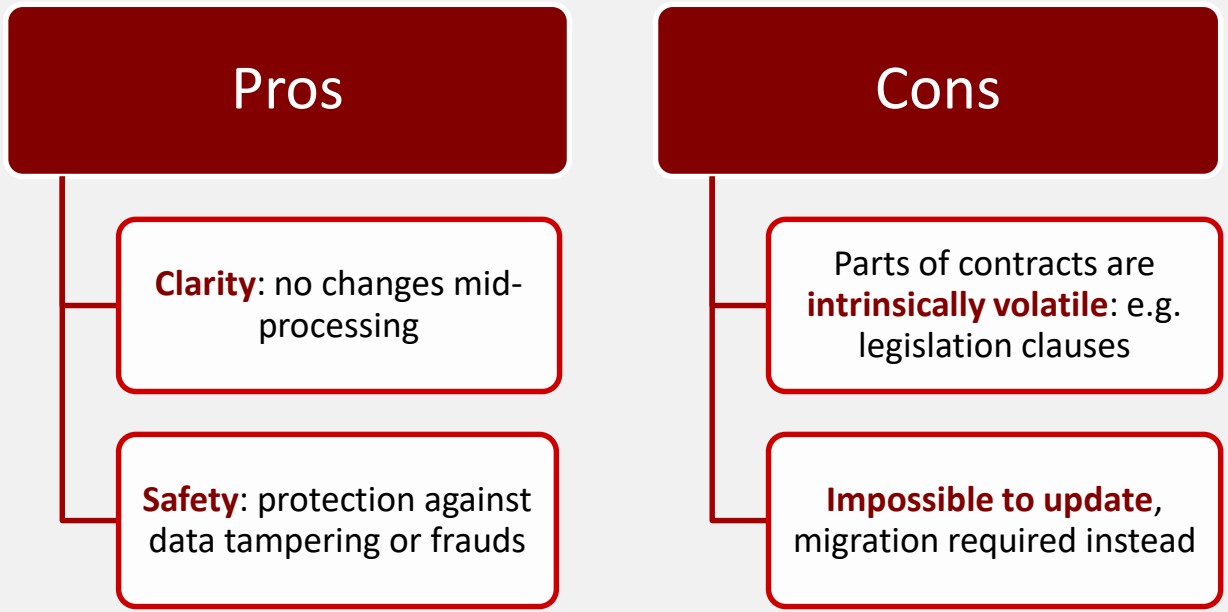


REAL WORLD CONTRACTS ARE COMPLICATED

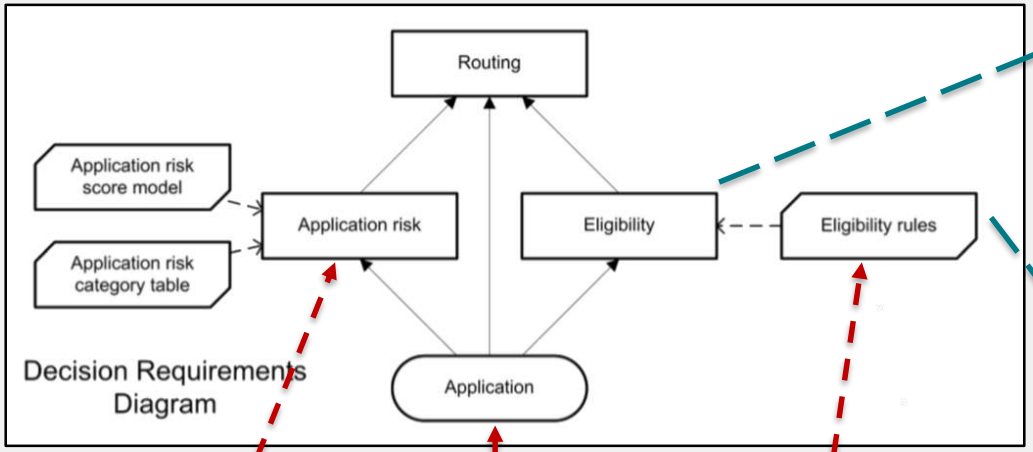
- Typically hundreds of clauses, not including legislation references
- Typically make assumptions that, in case of problems, have to be tested in court
- Translating them into a programming language (like Solidity) requires formalization of all scenarios

```
contract NameReg {

  mapping (address => string32) toName;
  mapping (string32 => address) toAddress;

  function register(string32 name) {
    // Don't allow the same name to be overwritten.
    if (toAddress[name] != address(0))
      return;
    // Unregister previous name if there was one.
    if (toName[msg.sender] != "")
      toAddress[toName[msg.sender]] = 0;

    toName[msg.sender] = name;
    toAddress[name] = msg.sender;
    AddressRegistered(msg.sender);
  }

  function unregister() {
    string32 n = toName[msg.sender];
    if (n == "")
      return;
    AddressDeregistered(toAddress[n]);
    toName[msg.sender] = "";
    toAddress[n] = address(0);
  }

  function addressOf(string32 name) constant returns (address addr) {
    return toAddress[name];
  }

  function nameOf(address addr) constant returns (string32 name) {
    return toName[addr];
  }

}
```

SMART CONTRACTS ARE IMMUTABLE



DMN STANDARDIZES BUSINESS DECISIONS NOTATION



Eligibility	
Eligibility rules	
Employment status	Application.Applicant.Employment. Status
Country	Application.Applicant.Country
Age	years and months duration(Application.Applicant.Date of birth, Application.Date).years

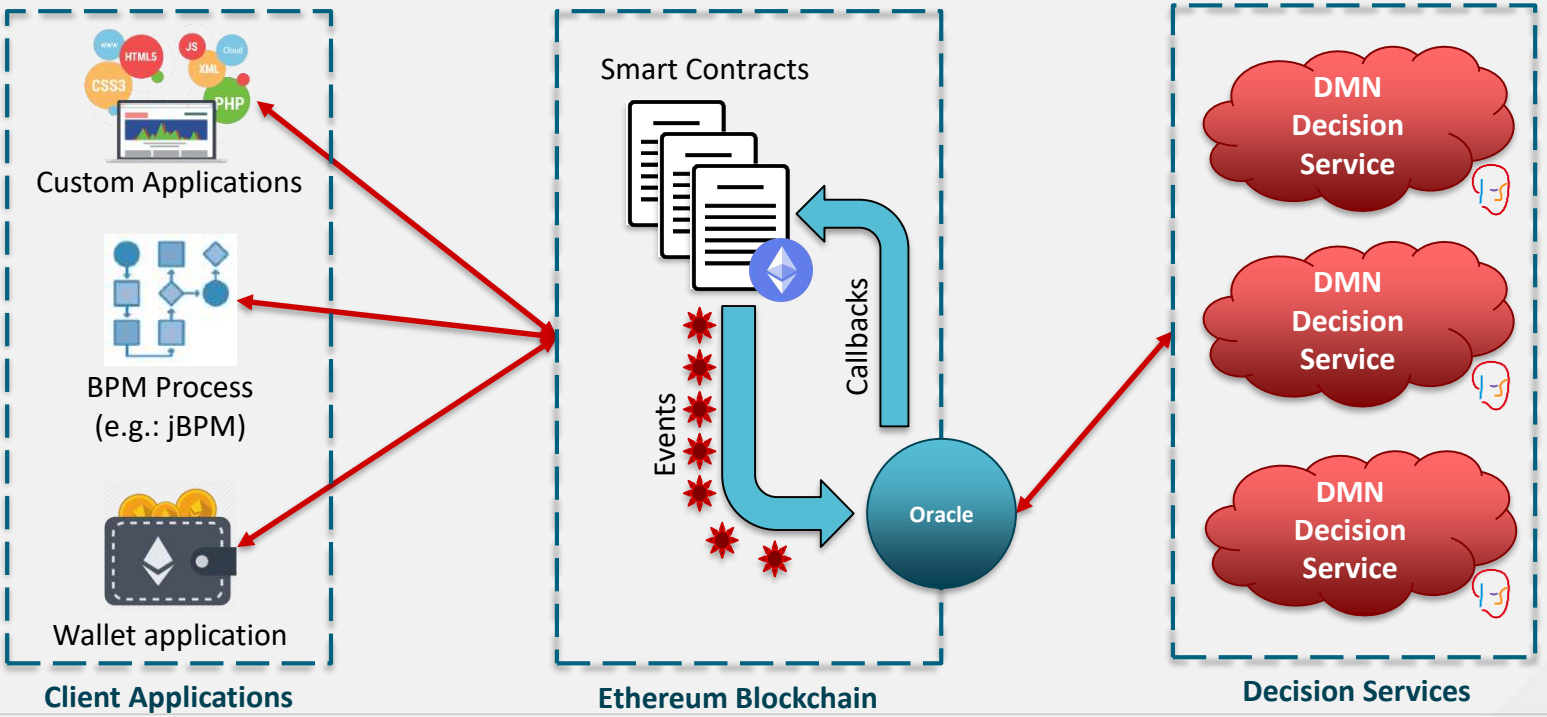
Eligibility rules				
P	Employment status	Country	Age	Eligibility
				INELIGIBLE, ELIGIBLE
1	*UNEMPLOYED*	-	-	*INELIGIBLE*
2	-	not(*UK*)	-	*INELIGIBLE*
3	-	-	< 18	*INELIGIBLE*
4	-	-	-	*ELIGIBLE*

Decision Node

Input Node

Business Knowledge Node

LEVERAGING DMN ON SMART CONTRACTS



MORE INFORMATION

Drools project: http://www.drools.org
Red Hat Decision Manager: https://www.redhat.com/en/technologies/jboss-middleware/decision-manager
Trisotech DMN Modeller: http://www.trisotech.com