SmartChainDB: Towards Semantic Events on Blockchains for Smart Manufacturing

Abhisha Bhattacharyya, Rahul Iyer, **Kemafor Anyanwu**

October 26, 2019

This work is funded by NSF grant CNS-1764025

Introduction

- Blockchains are increasingly being appropriated for different business processes
- Business processes typically comprise of different kinds of transactions, events and actors
- Beyond the need to support codifying of complex transactional processes, there is also a need to enable actors be notified about business process events relevant to them.

This work is funded by NSF grant CNS-1764025

Smart Contracts

- Mainstream blockchain platforms provide two core first-class primitives that support the ability to `create` and `transfer` assets
- Beyond the steps of creating and transferring assets, the rest of business process behavior is ”hidden” in programs referred to as `Smart Contracts.`
    - immutable blocks of code that when deployed on a blockchain executes automatically when certain conditions are met

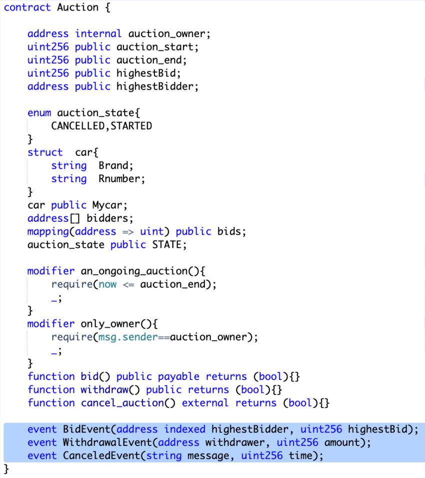
This work is funded by NSF grant CNS-1764025

Blockchains – Smart Contracts

- Steps in a business process, other than creating and transferring assets, are encoded as functions e.g. `bid, withdraw..`, etc.
- These functions emit ”events” (represented as strings in an event log) that can be consumed by the external environment
- From the point of view of usability, there are three key things that target users need from platforms managing smart contracts:
  - to be able to discover their existence
  - to be able to understand their behavior
  - to know when interesting application events have occurred during their execution

This work is funded by NSF grant CNS-1764025

An Example Smart Contract



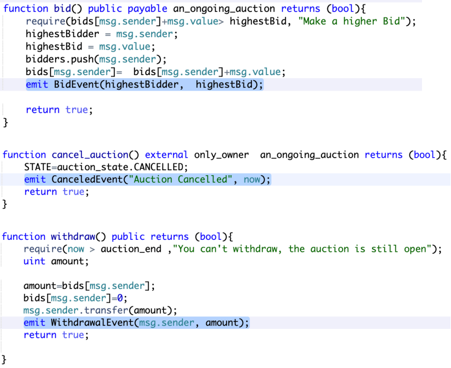
```
contract Auction {

    address internal auction_owner;
    uint256 public auction_start;
    uint256 public auction_end;
    uint256 public highestBid;
    address public highestBidder;

    enum auction_state{
        CANCELLED,STARTED
    }
    struct  car{
        string  Brand;
        string  Rnumber;
    }
    car public Mycar;
    address[] bidders;
    mapping(address => uint) public bids;
    auction_state public STATE;

    modifier an_ongoing_auction(){
        require(now <= auction_end);
        _;
    }
    modifier only_owner(){
        require(msg.sender==auction_owner);
        _;
    }
    function bid() public payable returns (bool){}
    function withdraw() public returns (bool){}
    function cancel_auction() external returns (bool){}

    event BidEvent(address indexed highestBidder, uint256 highestBid);
    event WithdrawalEvent(address withdrawer, uint256 amount);
    event CanceledEvent(string message, uint256 time);
}
```



```
function bid() public payable an_ongoing_auction returns (bool){
    require(bids[msg.sender]+msg.value> highestBid, "Make a higher Bid");
    highestBidder = msg.sender;
    highestBid = msg.value;
    bidders.push(msg.sender);
    bids[msg.sender]=  bids[msg.sender]+msg.value;
    emit BidEvent(highestBidder,  highestBid);

    return true;
}

function cancel_auction() external only_owner  an_ongoing_auction returns (bool){
    STATE=auction_state.CANCELLED;
    emit CanceledEvent("Auction Cancelled", now);
    return true;
}

function withdraw() public returns (bool){
    require(now > auction_end ,"You can't withdraw, the auction is still open");
    uint amount;

    amount=bids[msg.sender];
    bids[msg.sender]=0;
    msg.sender.transfer(amount);
    emit WithdrawalEvent(msg.sender, amount);
    return true;

}
```

This work is funded by NSF grant CNS-1764025

Motivation(1)

- Several issues with the current model
  - Anyone who wants to setup an auction has to implement auction contract with perhaps similar behavior
  - Potential consumers/customers will need to understand the terms of contract implemented in code
  - But how does one find out that there is an auction contract of interest in the first place? e.g. `auction for an automobile by their preferred manufacturer`
  - It may be possible to implement a publish/subscribe model based on the data in the event log, buts it usability will be limited by the well known "keyword search problem"

This work is funded by NSF grant CNS-1764025

Motivation(2): Need for More Complex Event Detection

- Consider a CyberManufacturing scenario (being made increasingly popular by the increasing complexity of manufacturing and technologies like 3D-printing)
    - Manufacturers may want to post `requests for quotes` for manufacturing capabilities
    - Potential suppliers will need to be made aware of requests for quotes that match their capabilities
- Even simple manufacturing capability description requests cannot be suitably represented with just textual phrases
    - terminological differences will present the first obstacle e.g. additive manufacturing vs. 3D-printing
    - descriptions may be implicit and may require some semantic mapping, e.g. **If Material = PolyCarbonate and Quantity < 10 Then Capability is 3D-Printing**
- Existing topic-based publish/subscribe models that are based on keyword topic labels will not be adequate

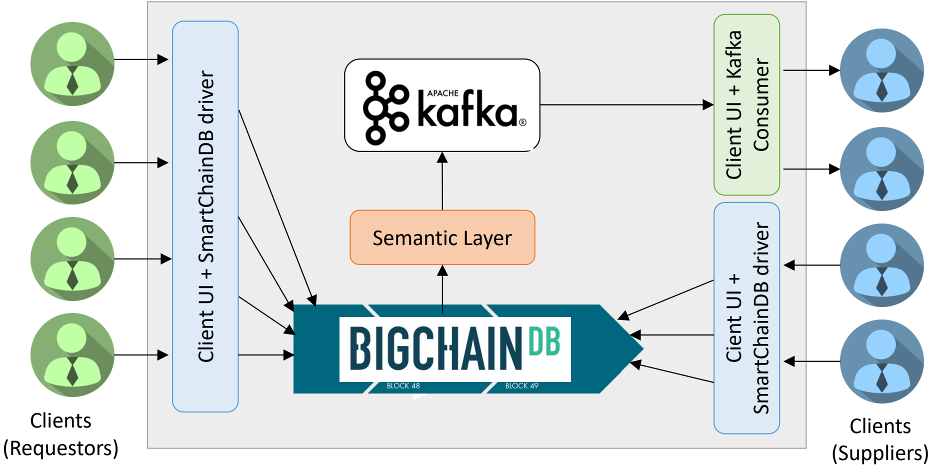
This work is funded by NSF grant CNS-1764025

Our Position

- It is necessary to introduce additional primitives that will support a broader range of business process steps to be captured declaratively e.g. `Request For Quote, Bid, ..`
  - A major success factor for relational database systems!!
- Need to support semantic technologies e.g. ontologies and ontological reasoning integrated with publish/subscribe models
- Additional requirements such as privacy
- These ideas are being implemented in a project called `SmartChainDB`
  - SmartChainDB builds on the BigChainDB blockchain platform
  - BigChainDB's architecture is more amenable to the nature of extensibility desired

This work is funded by NSF grant CNS-1764025

SmartChainDb – Architecture



This work is funded by NSF grant CNS-1764025

Conclusion

- Work is ongoing a
- Contact: kogan@ncsu.edu
- **Acknowledgements**
  - CoPIs - Binil Starly (Professor of Industrial System Engineering), Alessandra Scafuro (Assistant Professor of Computer Science)
  - Funding: National Science Foundation

This work is funded by NSF grant CNS-1764025