SAPIENZA
UNIVERSITÀ DI ROMA

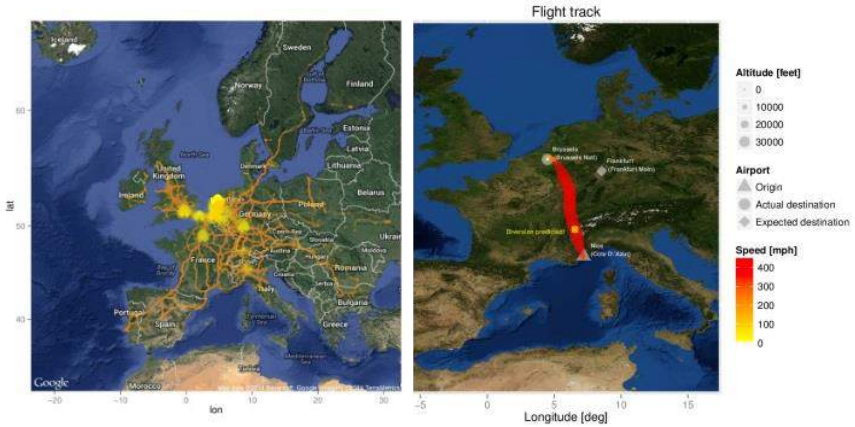
Blockchain as a process execution infrastructure

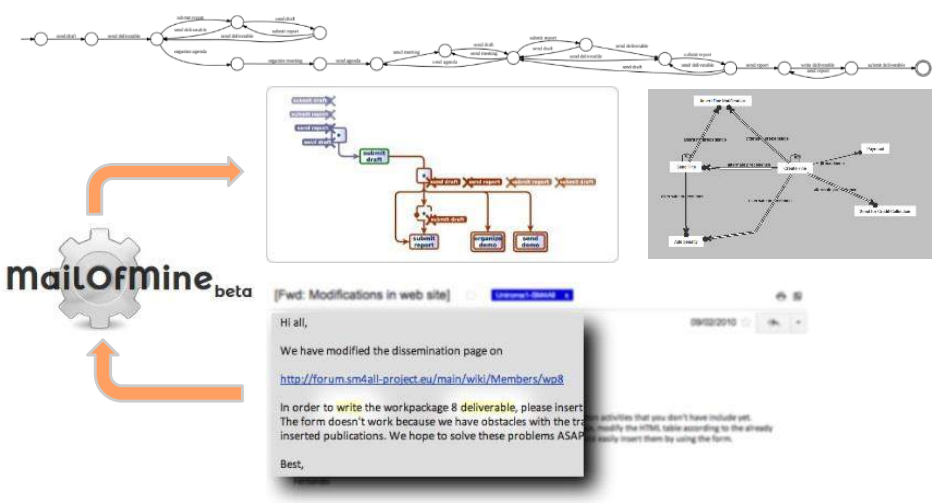
Claudio Di Ciccio | diciccio.net | claudio.diciccio@uniroma1.it
Assistant Professor, Sapienza University of Rome, Italy | uniroma1.it

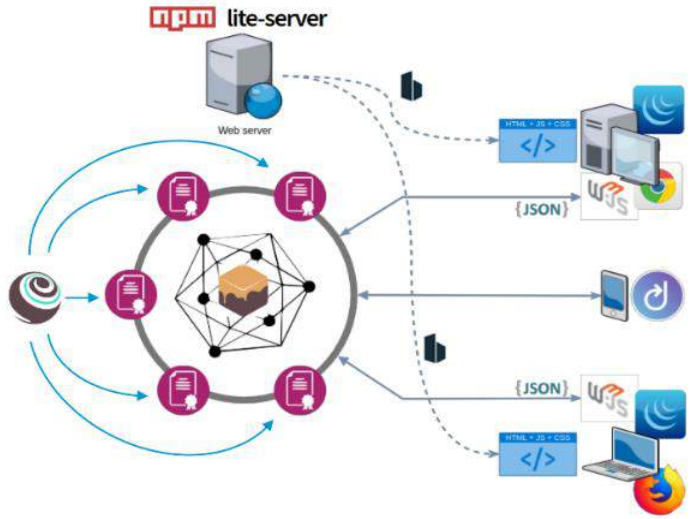
Claudio Di Ciccio

Assistant professor
Ph.D. in Computer Science and Engineering

Main research interests:
process mining,
blockchains,
declarative process modelling,
service-oriented architectures







My experience so far

Latina, Italy (B.Sc)

Rome, Italy (M.Sc, Ph.D)

Vienna, Austria (Post-doc, Assistant Prof.)

Rome, Italy (Assistant Prof.)

Half empty or half full?

Which is more fundamental: processes or things?

Neither half-full nor half-empty. *Courtesy Wikipedia*

http://aeon.co/ideas/which-is-more-fundamental-processes-or-things

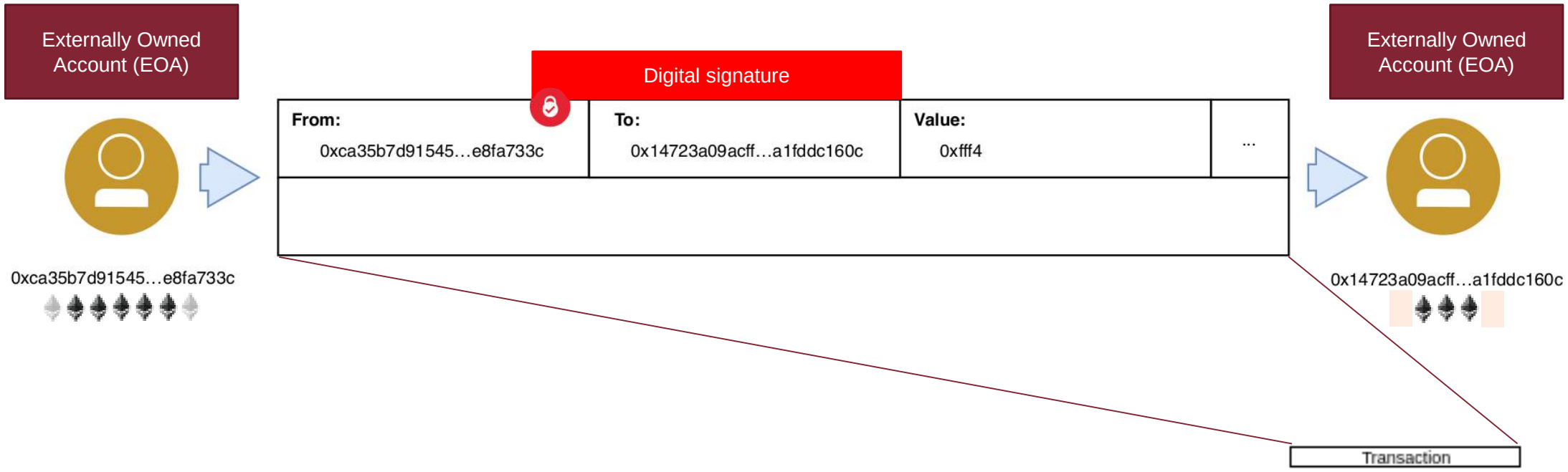
Processes are into dynamics

Thanks Dr. Mieke Jans for offering her hand

Blockchain as an infrastructure

Transaction

- Transfer of **(crypto)assets** (Ether, Bitcoin, Algo, …) from **account** A to **account** B



Metaphor

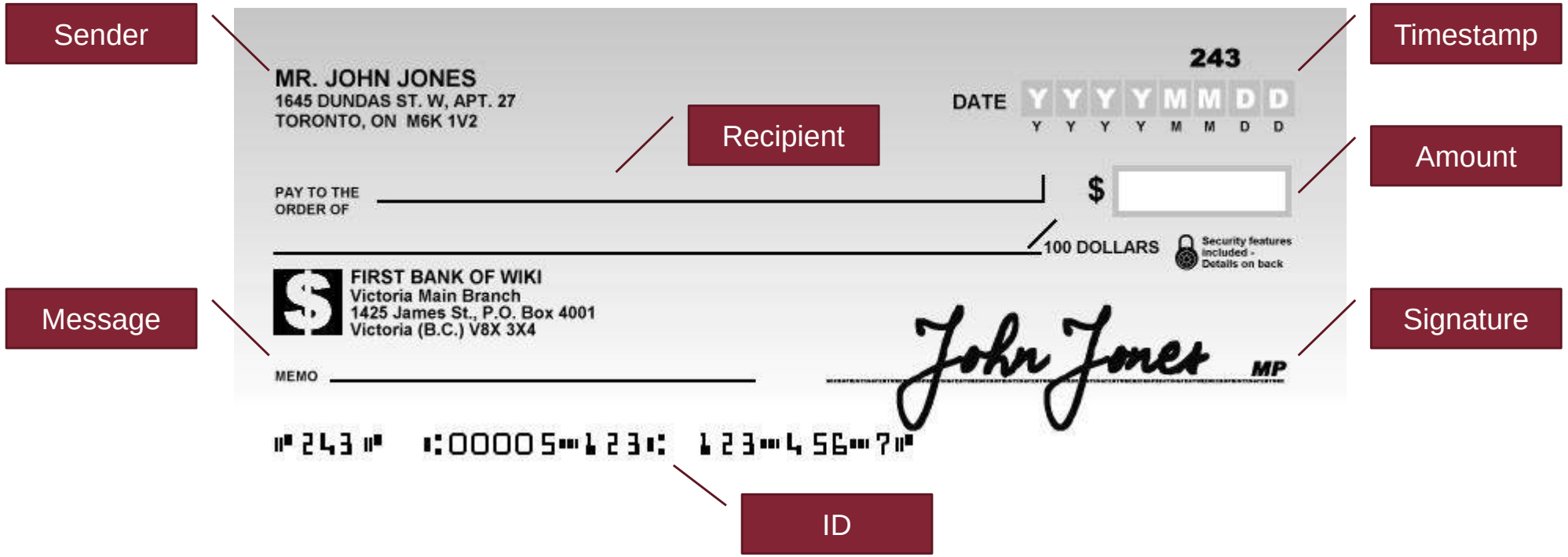
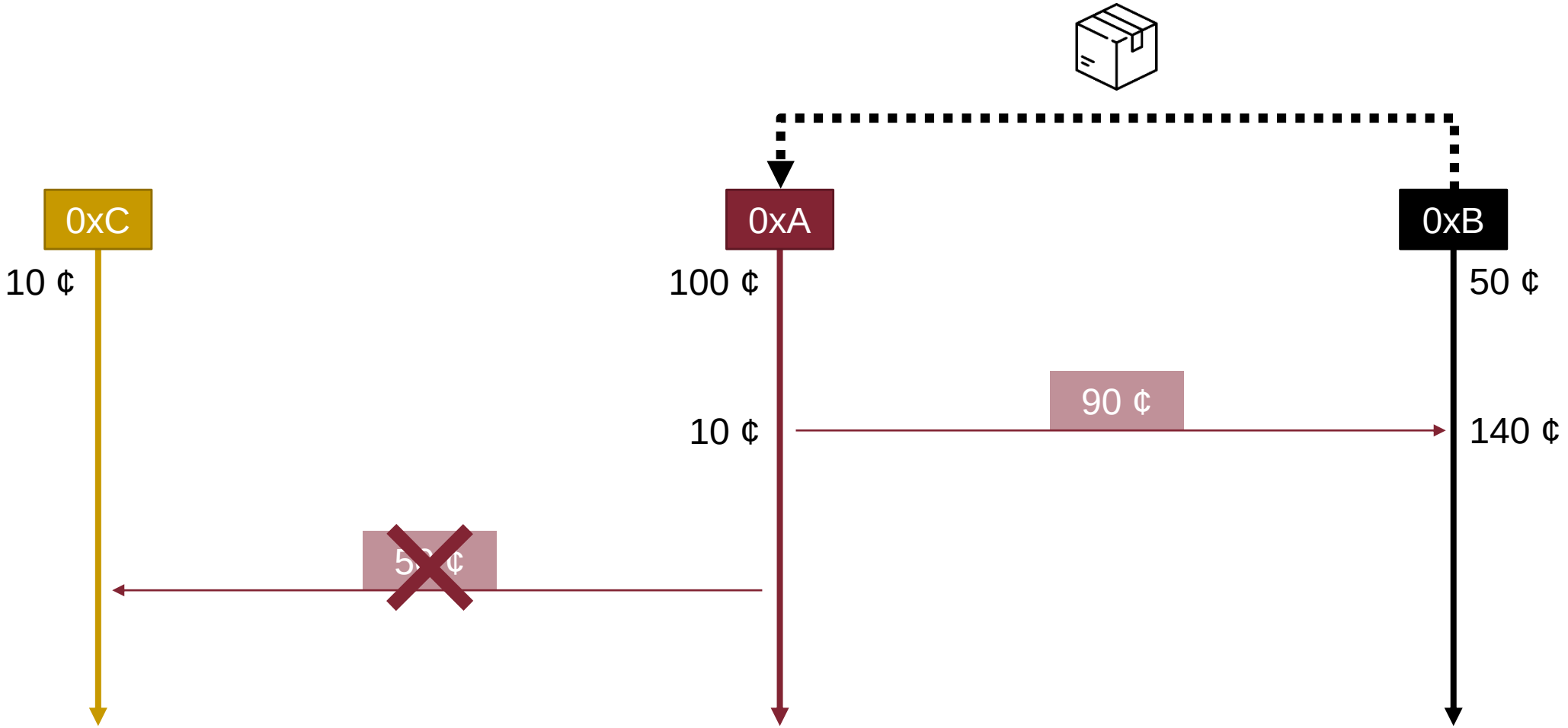


Image by Airodyssey, https://en.wikipedia.org/wiki/File:BlankCheque.png

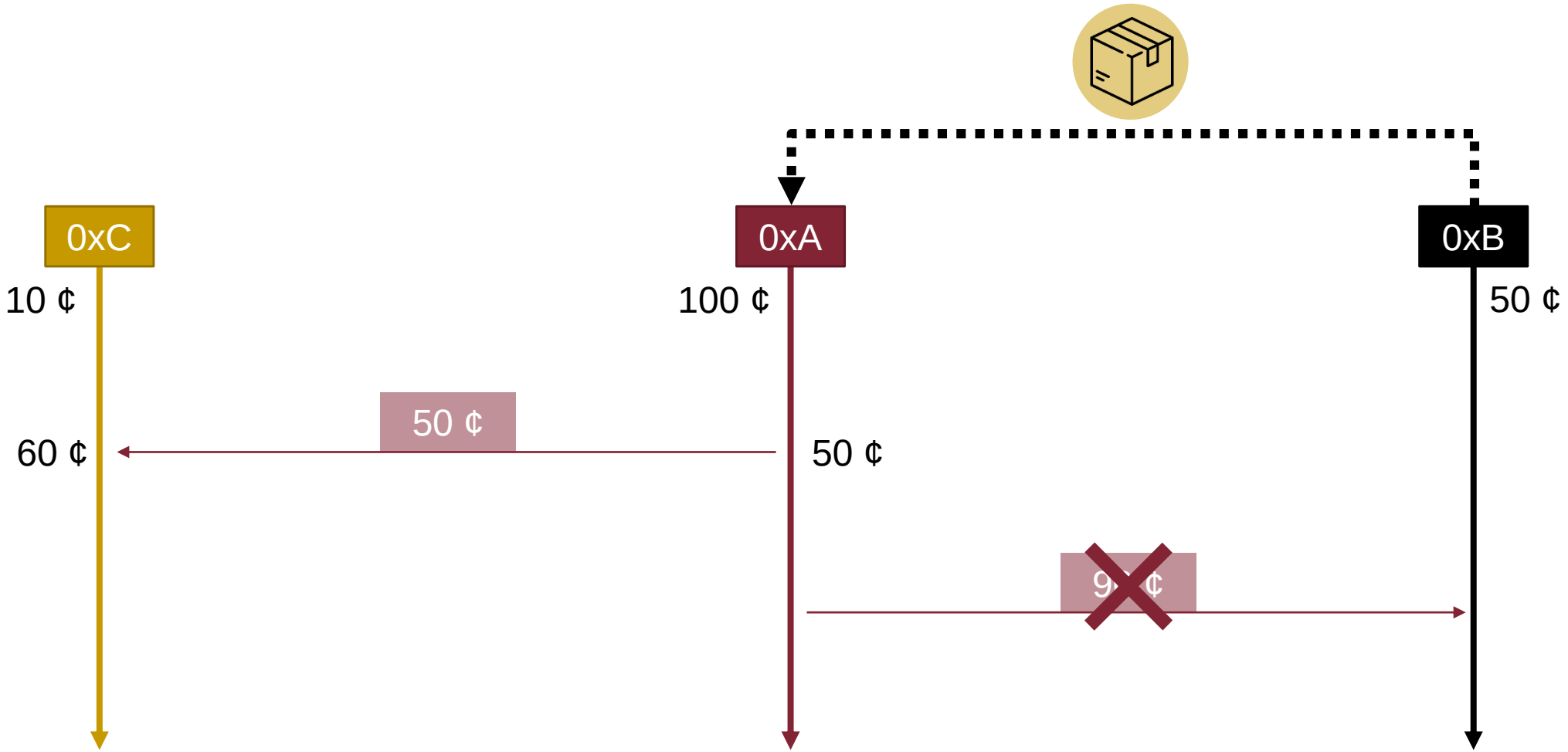
Ledger

- Ordered collection of transactions
- The **order** matters!

Double spending

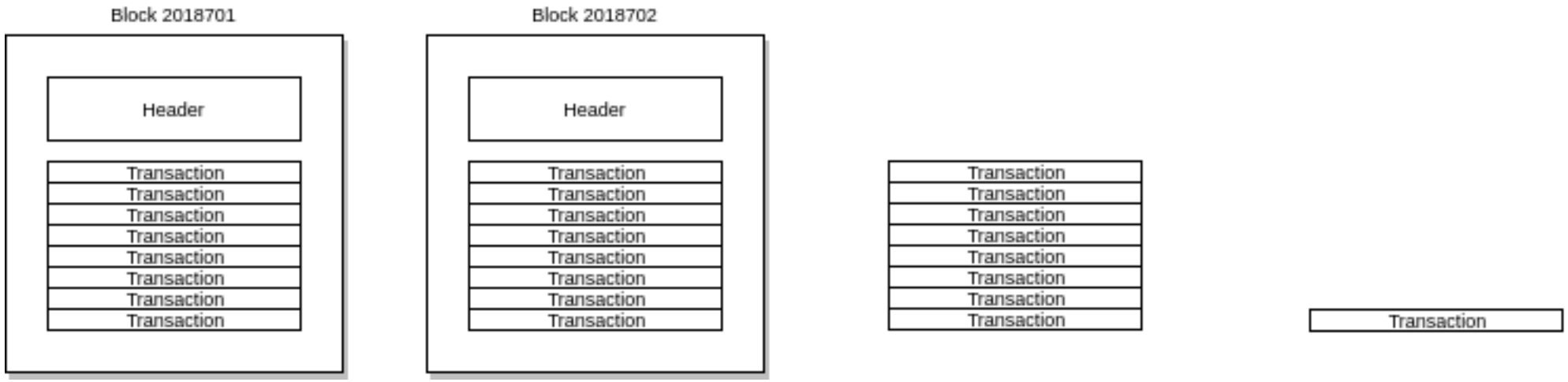


Double spending



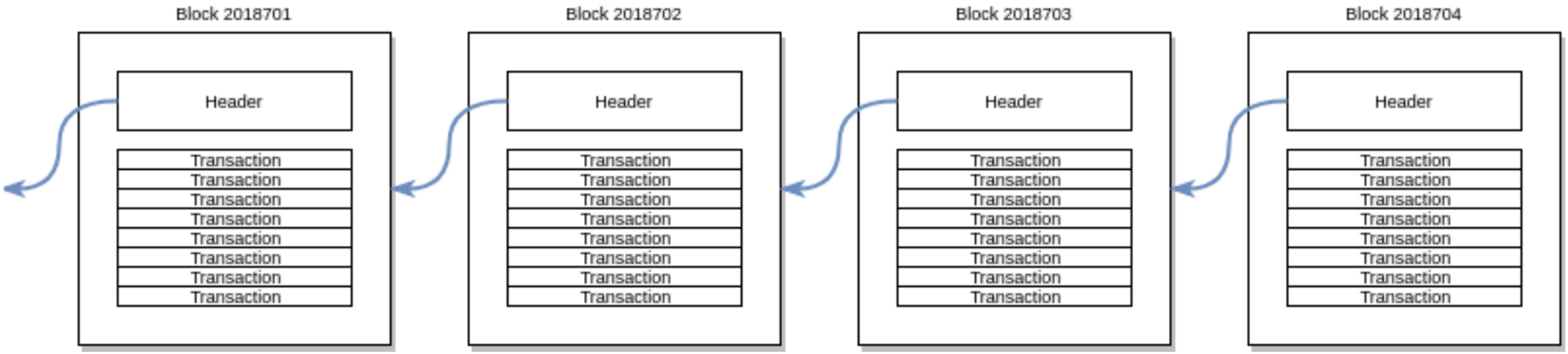
Block

- Blocks group and collate transactions
- The order matters!

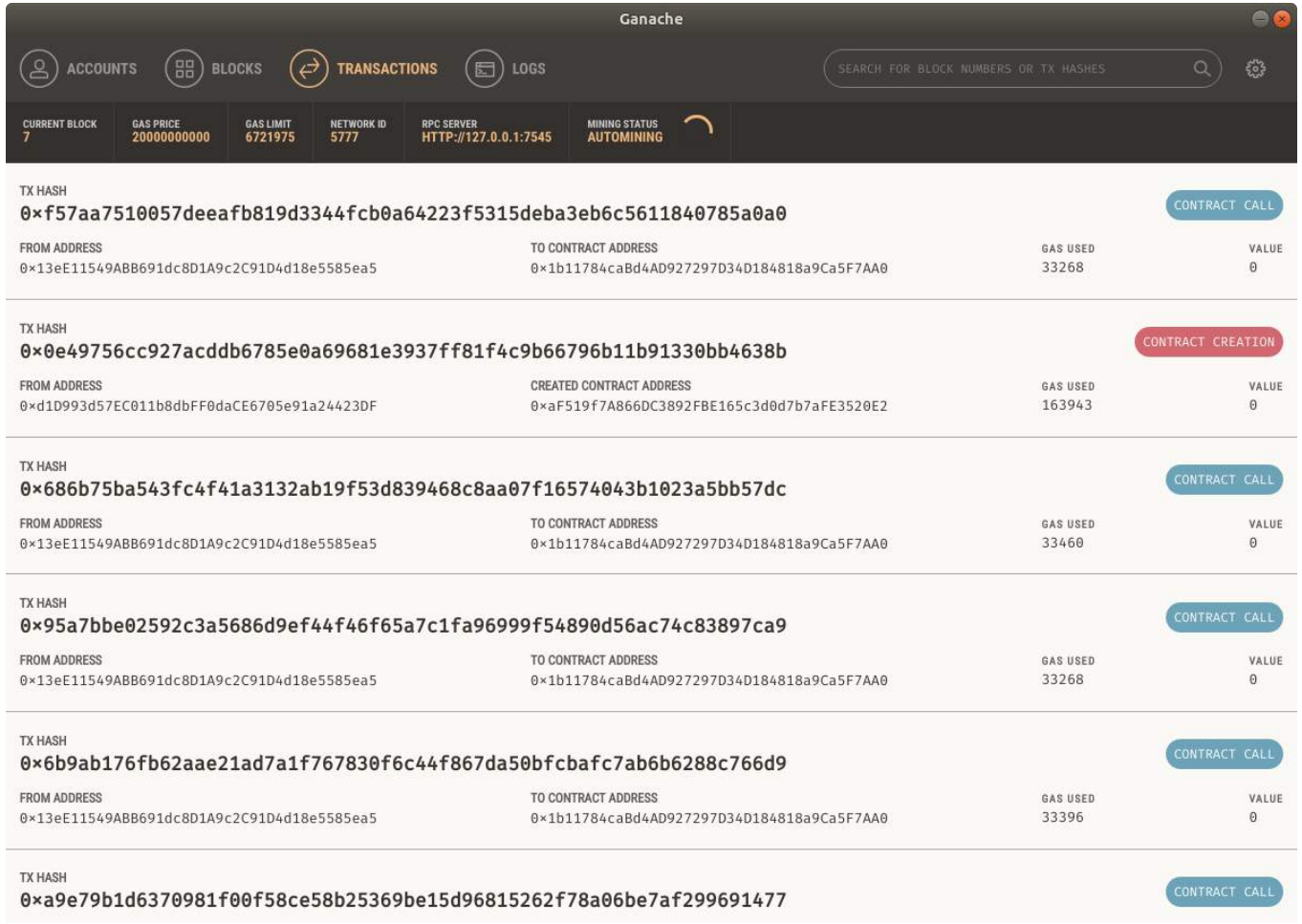


Hashing the previous block for immutability

- Blocks refer back to direct predecessors via **hashing**
- The order matters!



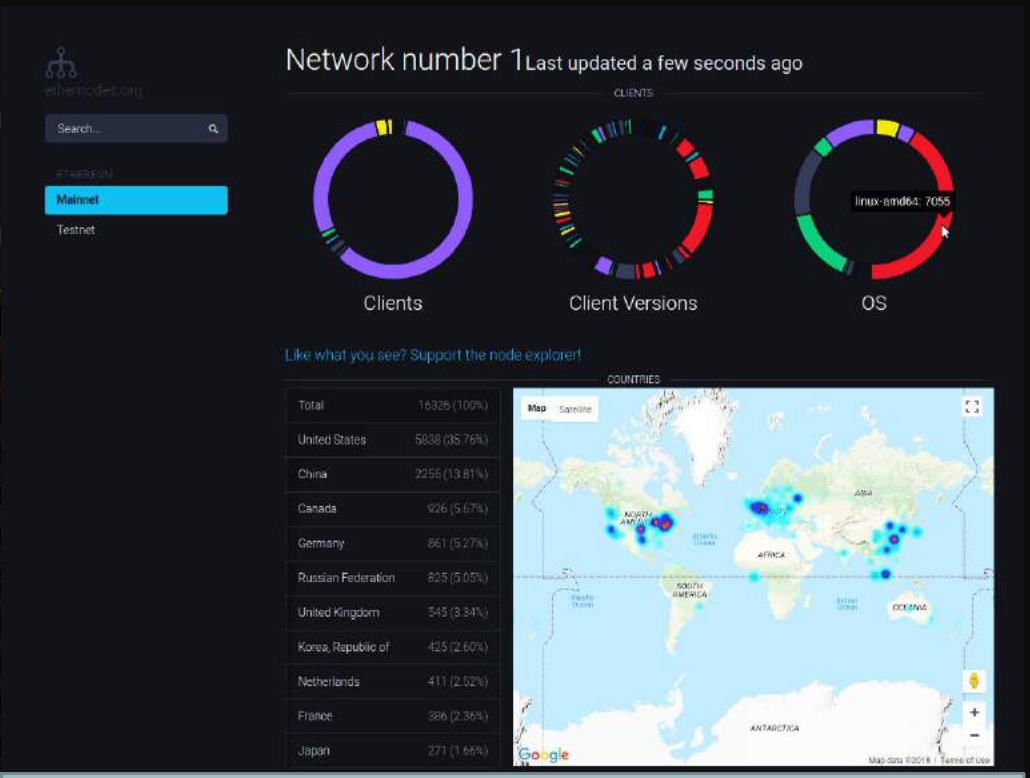
The blockchain remembers



Shortcomings of centralised ledgers

- Potentially
  - lost or destroyed
  - containing invalid transactions
  - incomplete
  - altered

Ledgers are distributed and maintained by a network

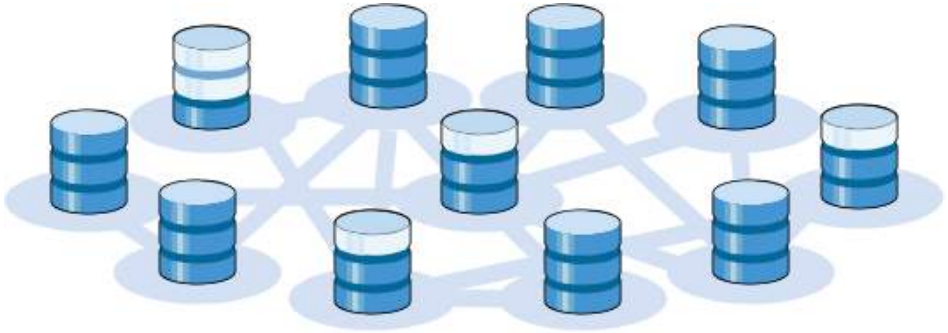


https://ehtstats.net

https://ehternodes.org

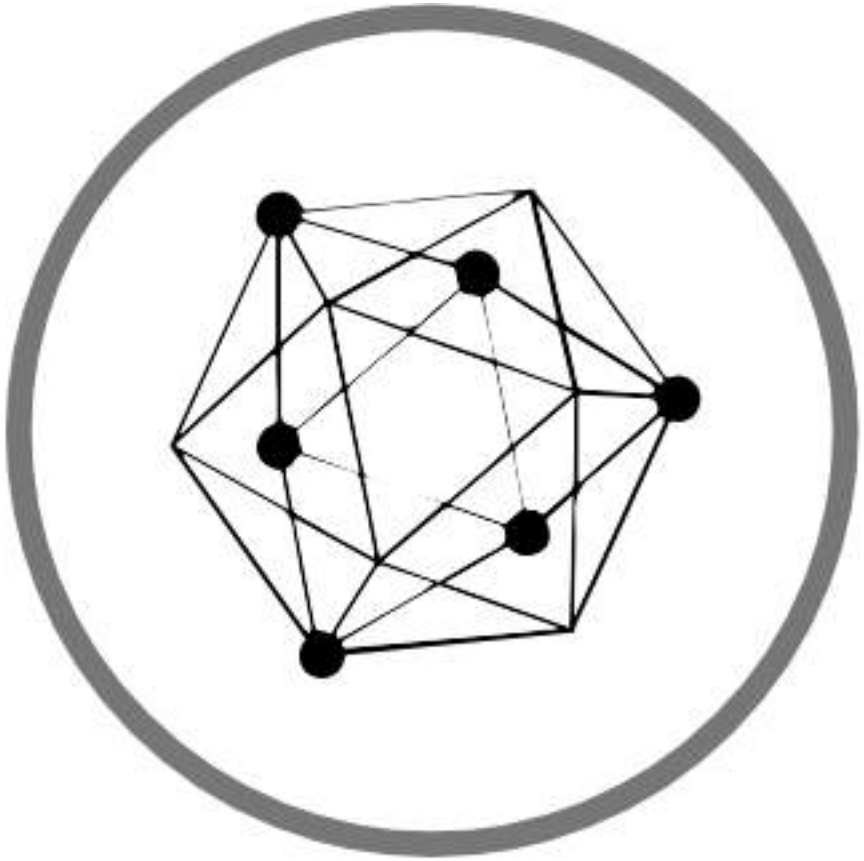
Decentralisation for persistence

Centralisation	Decentralisation

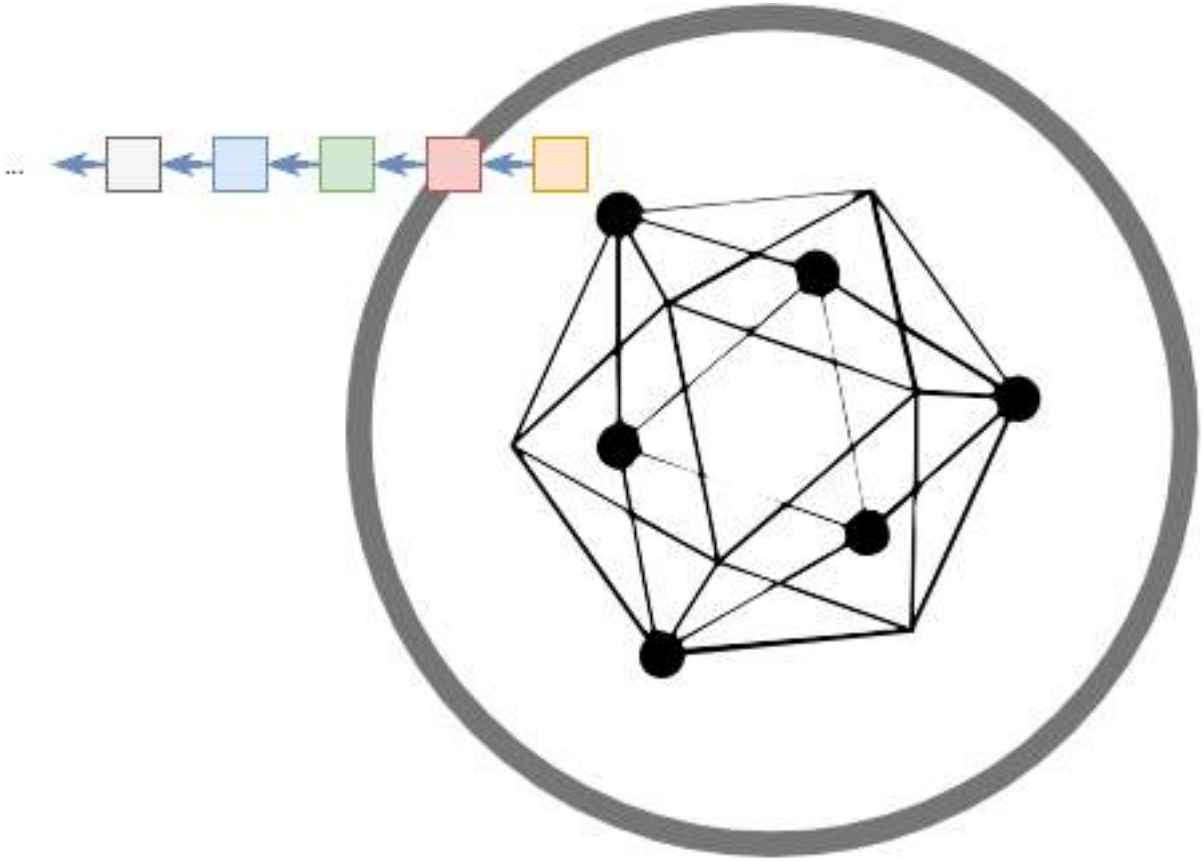


Warning: possible information inconsistency → mining and consensus

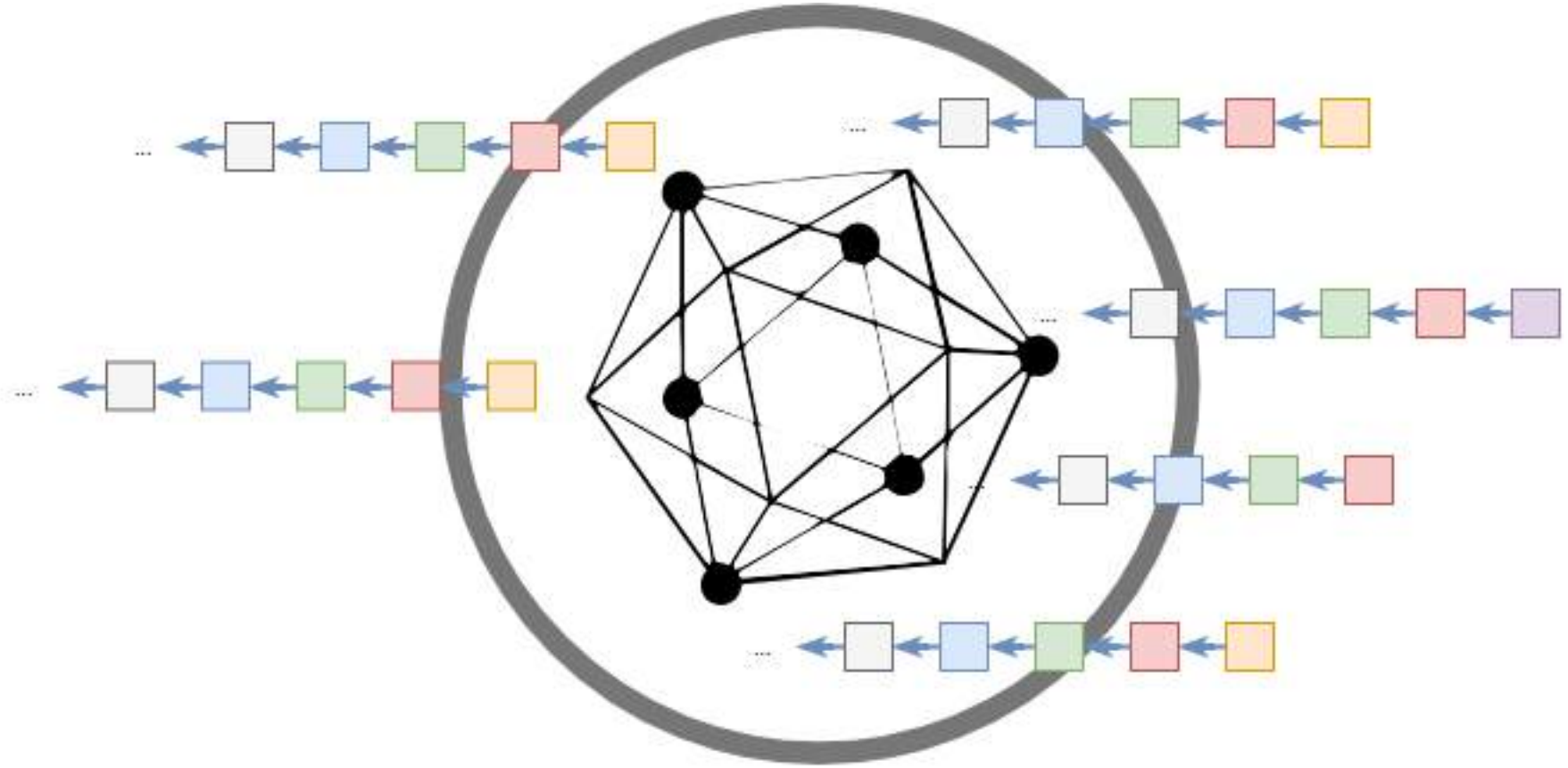
Distributed nature



Distributed nature

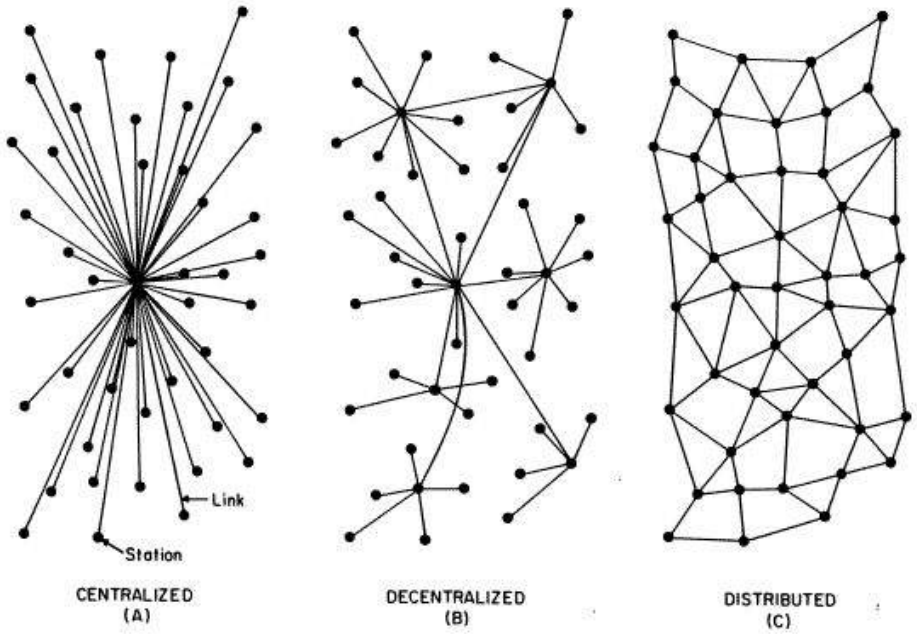


Distributed nature



Centralized, decentralized, distributed

- Politically **decentralised**
  - No entity controls the network
- Logically **centralised**
  - One commonly agreed state*
- Architecturally **distributed**
  - Nodes have access to the full history



Mining for a blockchain

Proof of Work (PoW): obtain the **right to publish the next block** by solving a **computationally intensive puzzle**

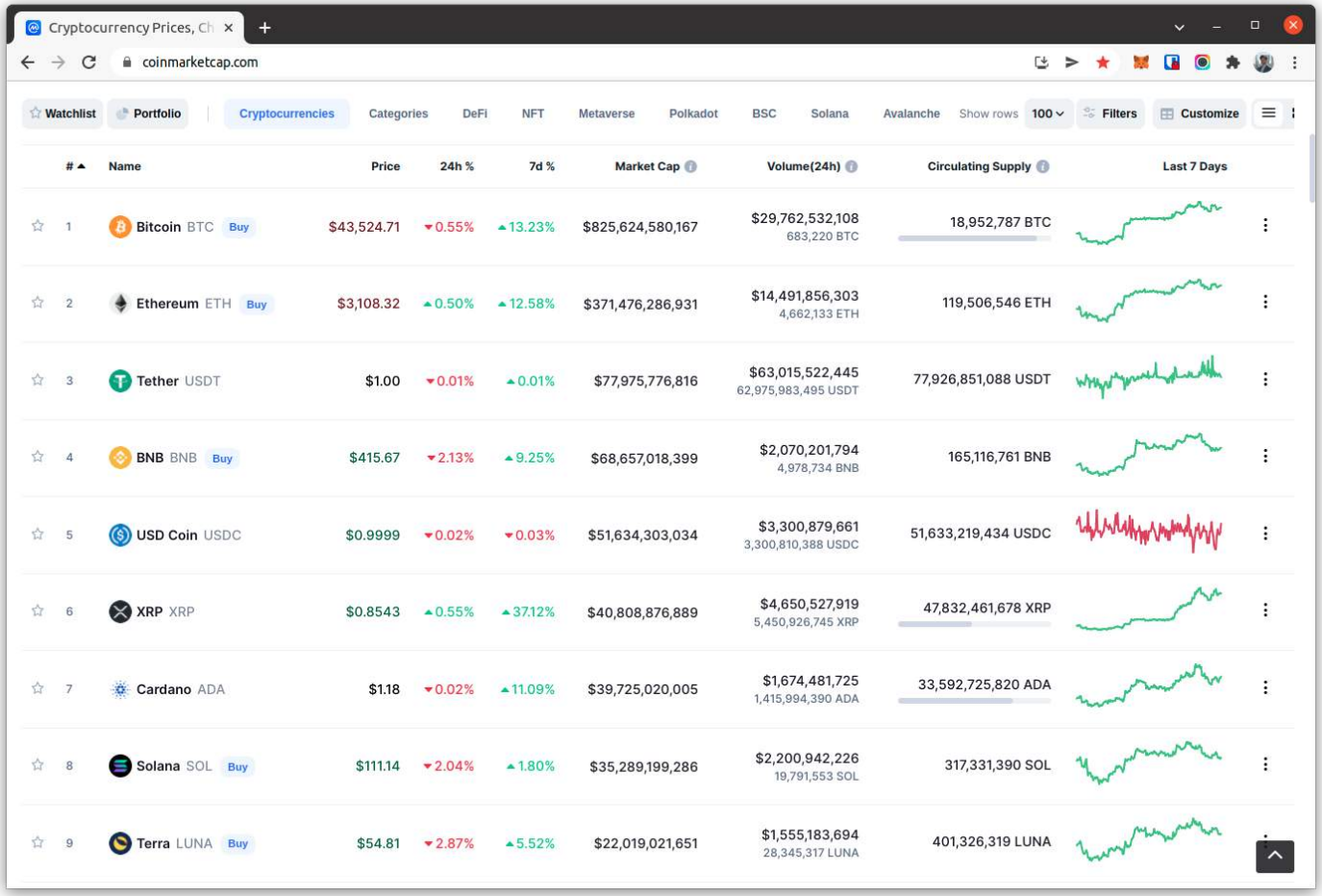
Proof of Stake (PoS): decide on the **next block** to be published and put **cryptoassets at stake** for it

Checking that a **solution** is **valid** is **easy**

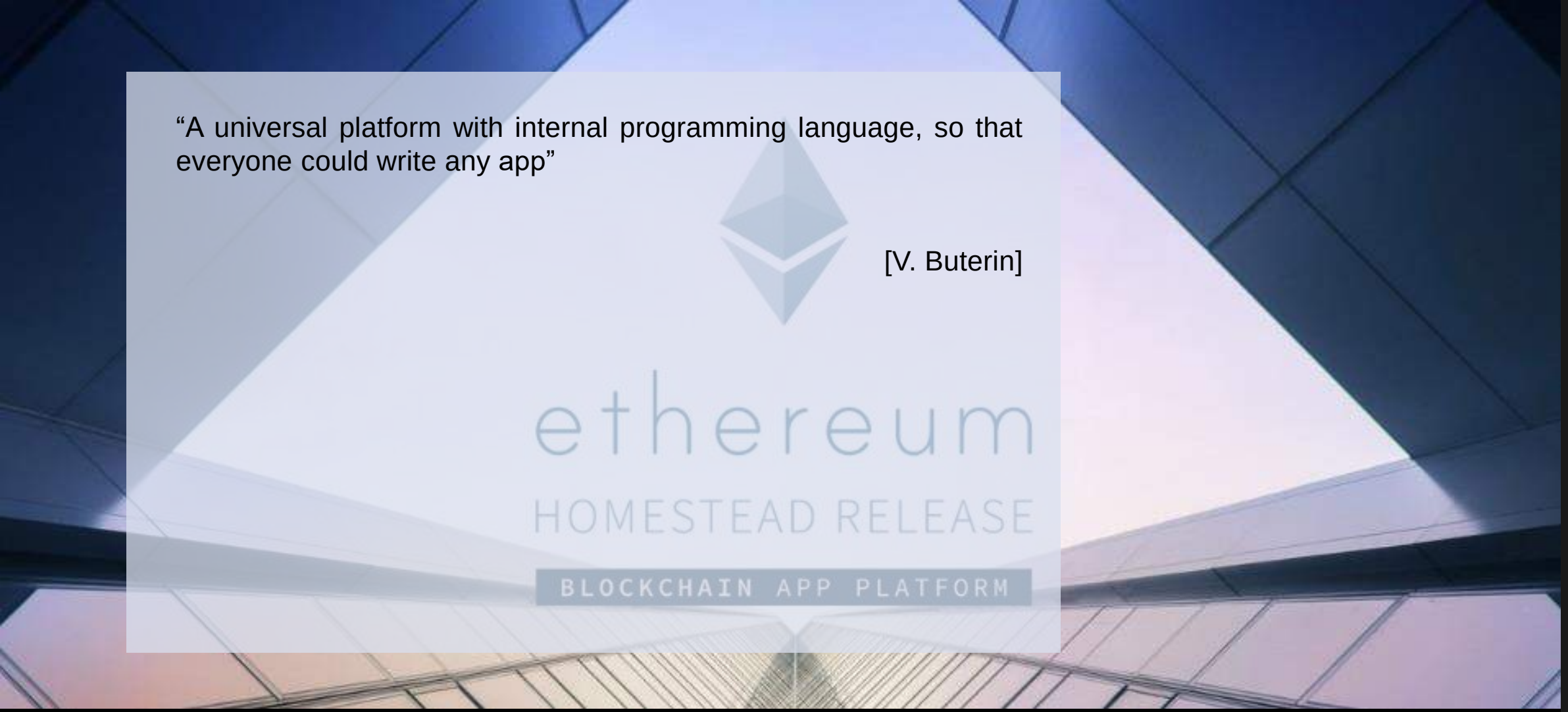
An **incentive** is needed to keep the infrastructure up

Crypto-fuel needed!

Crypto-fuel needed



#	Name	Price	24h %	7d %	Market Cap	Volume(24h)	Circulating Supply	Last 7 Days
1	Bitcoin BTC Buy	$43,524.71	0.55%	13.23%	$825,624,580,167	$29,762,532,108 683,220 BTC	18,952,787 BTC	
2	Ethereum ETH Buy	$3,108.32	0.50%	12.58%	$371,476,286,931	$14,491,856,303 4,662,133 ETH	119,506,546 ETH	
3	Tether USDT	$1.00	0.01%	0.01%	$77,975,776,816	$63,015,522,445 62,975,983,495 USDT	77,926,851,088 USDT	
4	BNB BNB Buy	$415.67	2.13%	9.25%	$68,657,018,399	$2,070,201,794 4,978,734 BNB	165,116,761 BNB	
5	USD Coin USDC	$0.9999	0.02%	0.03%	$51,634,303,034	$3,300,879,661 3,300,810,388 USDC	51,633,219,434 USDC	
6	XRP XRP	$0.8543	0.55%	37.12%	$40,808,876,889	$4,650,527,919 5,450,926,745 XRP	47,832,461,678 XRP	
7	Cardano ADA	$1.18	0.02%	11.09%	$39,725,020,005	$1,674,481,725 1,415,994,390 ADA	33,592,725,820 ADA	
8	Solana SOL Buy	$111.14	2.04%	1.80%	$35,289,199,286	$2,200,942,226 19,791,553 SOL	317,331,390 SOL	
9	Terra LUNA Buy	$54.81	2.87%	5.52%	$22,019,021,651	$1,555,183,694 28,345,317 LUNA	401,326,319 LUNA	



From peer-to-peer electronic cash system to programmable distributed environment

```
pragma solidity ^0.4.0;

contract HelloToken {
    address public minter;
    mapping (address => uint) public balance;
    uint public constant PRICE = 2 finney;

    constructor() public {
        minter = msg.sender;
    }

    function mint() public payable {
        require(msg.value >= PRICE, "Not enough value for a token!");
        balance[msg.sender] += msg.value / 2 finney;
    }

    function transfer(uint amount, address to) public {
        require(balance[msg.sender] >= amount, "Not enough tokens!");
        balance[msg.sender] -= amount;
        balance[to] += amount;
    }

    function terminate() public {
        require(msg.sender == minter, "You cannot terminate the contract!");
        selfdestruct(minter);
    }
}
```

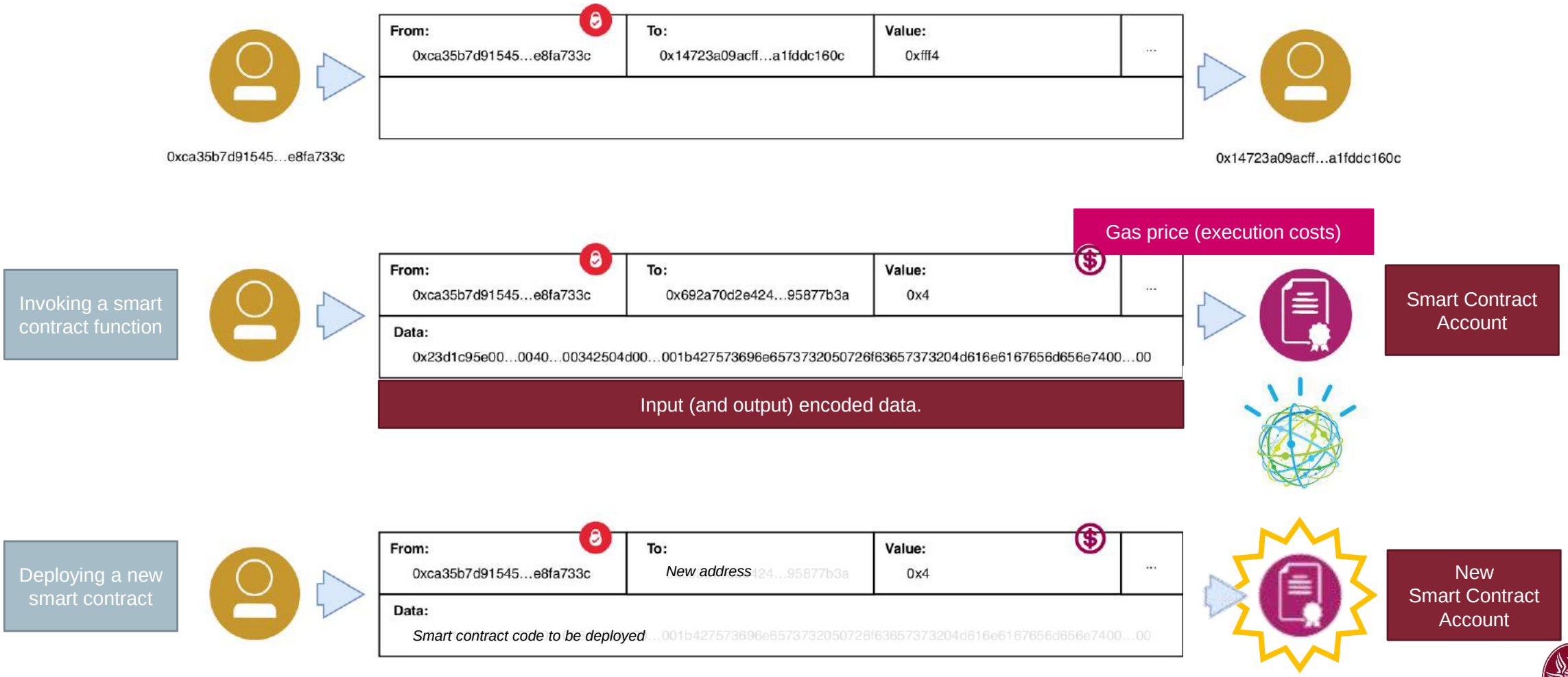
Smart contracts

Smart Contracts are pieces of code

```
pragma solidity ^0.4.0;

contract HelloToken {
    address public minter;
    mapping (address => uint) public balance;
    uint public constant PRICE = 2 finney;

    constructor() public {
        minter = msg.sender;
    }

    function mint() public payable {
        require(msg.value >= PRICE, "Not enough value for a token!");
        balance[msg.sender] += msg.value / 2 finney;
    }

    function transfer(uint amount, address to) public {
        require(balance[msg.sender] >= amount, "Not enough tokens!");
        balance[msg.sender] -= amount;
        balance[to] += amount;
    }

    function terminate() public {
        require(msg.sender == minter, "You cannot terminate the contract!");
        selfdestruct(minter);
    }
}

```

- Smart Contracts in Ethereum
  - live in the Ethereum environment
  - execute a function when called
  - have direct control over their own balance and key/value storage
  - have their behaviour fully specified by their **code**

A programmable distributed environment



The Ethereum Virtual Machine (EVM)

- Think of the EVM as an **emulation** of a **single, global "computer"**
- A **globally accessible virtual machine** (like a **mainframe**)
  - in fact, lots of smaller computers
- There is a cost associated to the running of programs on the EVM
  - Please welcome the **gas**

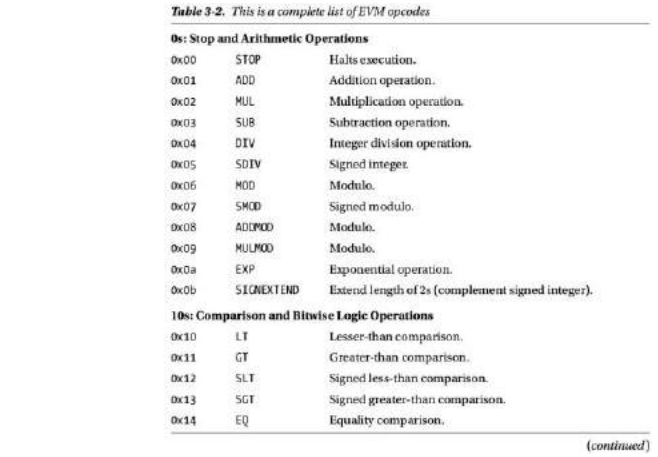
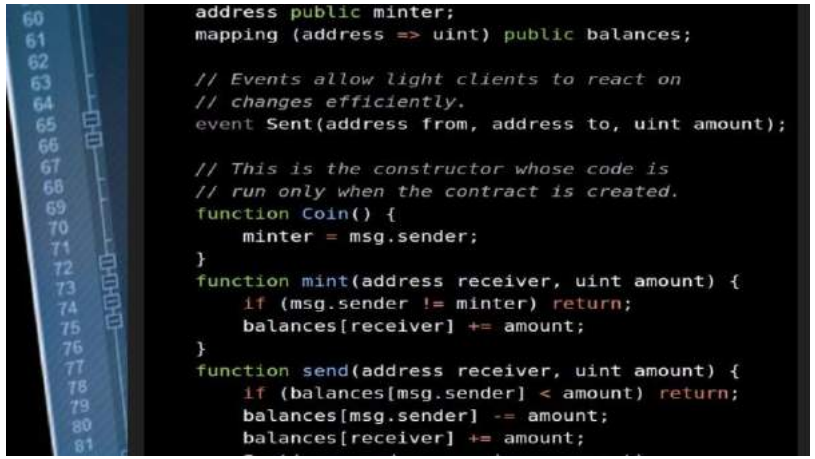


Table 3-2. *This is a complete list of EVM opcodes*

0s: Stop and Arithmetic Operations		
0x00	STOP	Halts execution.
0x01	ADD	Addition operation.
0x02	MUL	Multiplication operation.
0x03	SUB	Subtraction operation.
0x04	DIV	Integer division operation.
0x05	SDIV	Signed integer.
0x06	MOD	Modulo.
0x07	SMOD	Signed modulo.
0x08	ADDMOD	Modulo.
0x09	MULMOD	Modulo.
0x0a	EXP	Exponential operation.
0x0b	SIGNEXTEND	Extend length of 2s (complement signed integer).
10s: Comparison and Bitwise Logic Operations		
0x10	LT	Lesser-than comparison.
0x11	GT	Greater-than comparison.
0x12	SLT	Signed less-than comparison.
0x13	SGT	Signed greater-than comparison.
0x14	EQ	Equality comparison.

(continued)



```
    address public minter;
    mapping (address => uint) public balances;

    // Events allow light clients to react on
    // changes efficiently.
    event Sent(address from, address to, uint amount);

    // This is the constructor whose code is
    // run only when the contract is created.
    function Coin() {
        minter = msg.sender;
    }
    function mint(address receiver, uint amount) {
        if (msg.sender != minter) return;
        balances[receiver] += amount;
    }
    function send(address receiver, uint amount) {
        if (balances[msg.sender] < amount) return;
        balances[msg.sender] -= amount;
        balances[receiver] += amount;
```


Image on the top-right courtesy of BlockchainHub (blockchainhub.net). Image on the bottom-right corner: [IES]

The paradigm

Mainframe

Terminal

L: https://commons.wikimedia.org/wiki/File:SiemensHicom300_1200.jpg; R: https://commons.wikimedia.org/wiki/File:HP_2647A_terminal.jpg

Metaphor (updated)

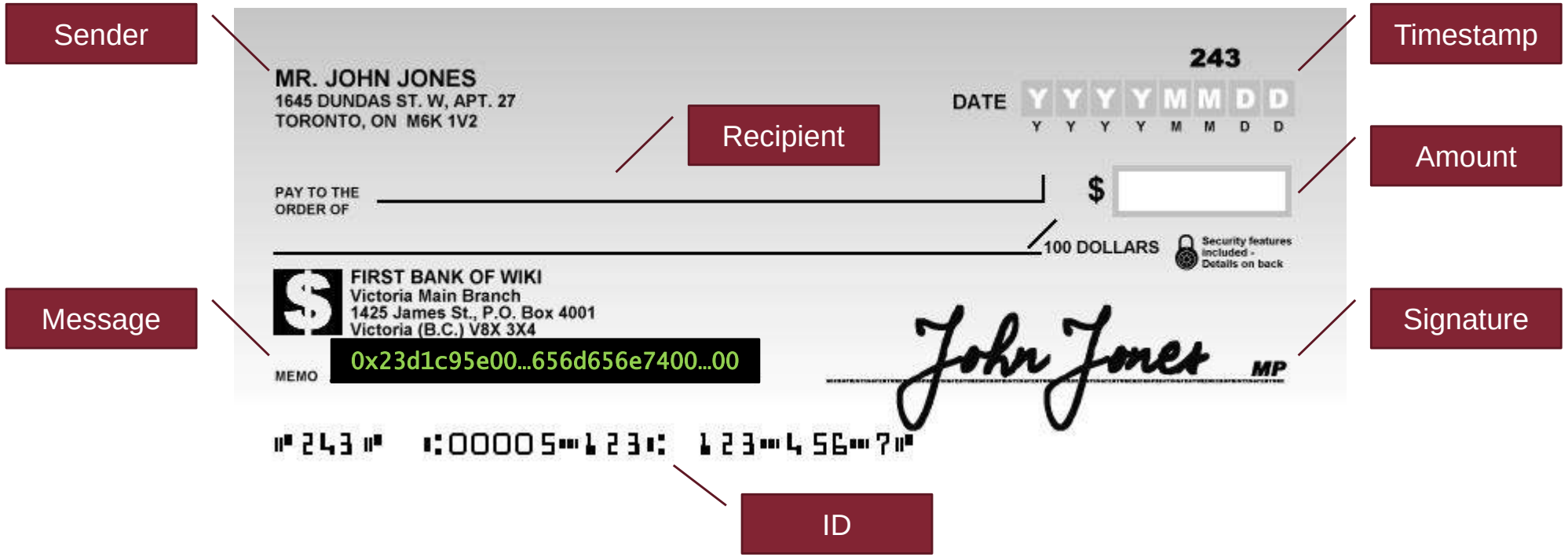


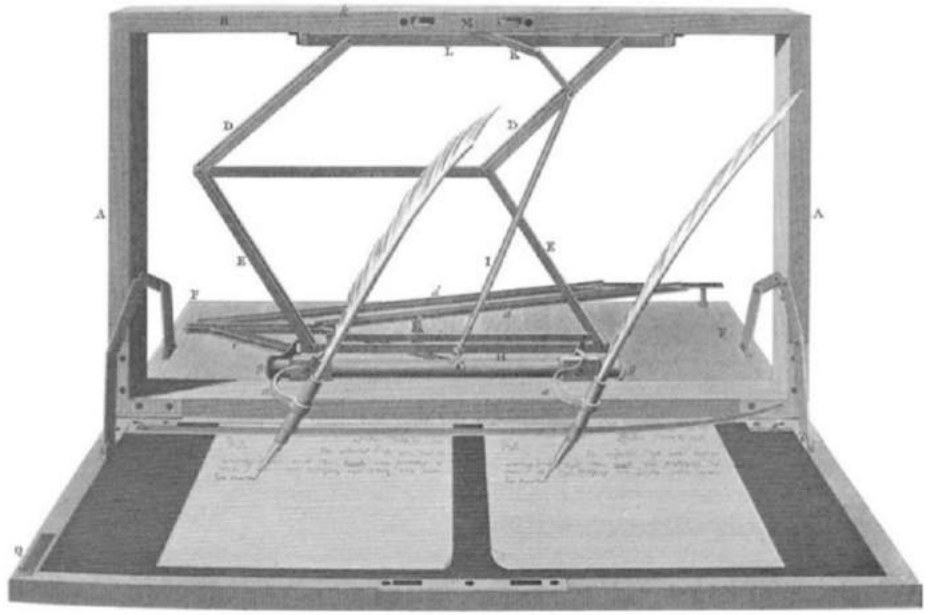
Image by Airodyssey, https://en.wikipedia.org/wiki/File:BlankCheque.png

The polygraph machine

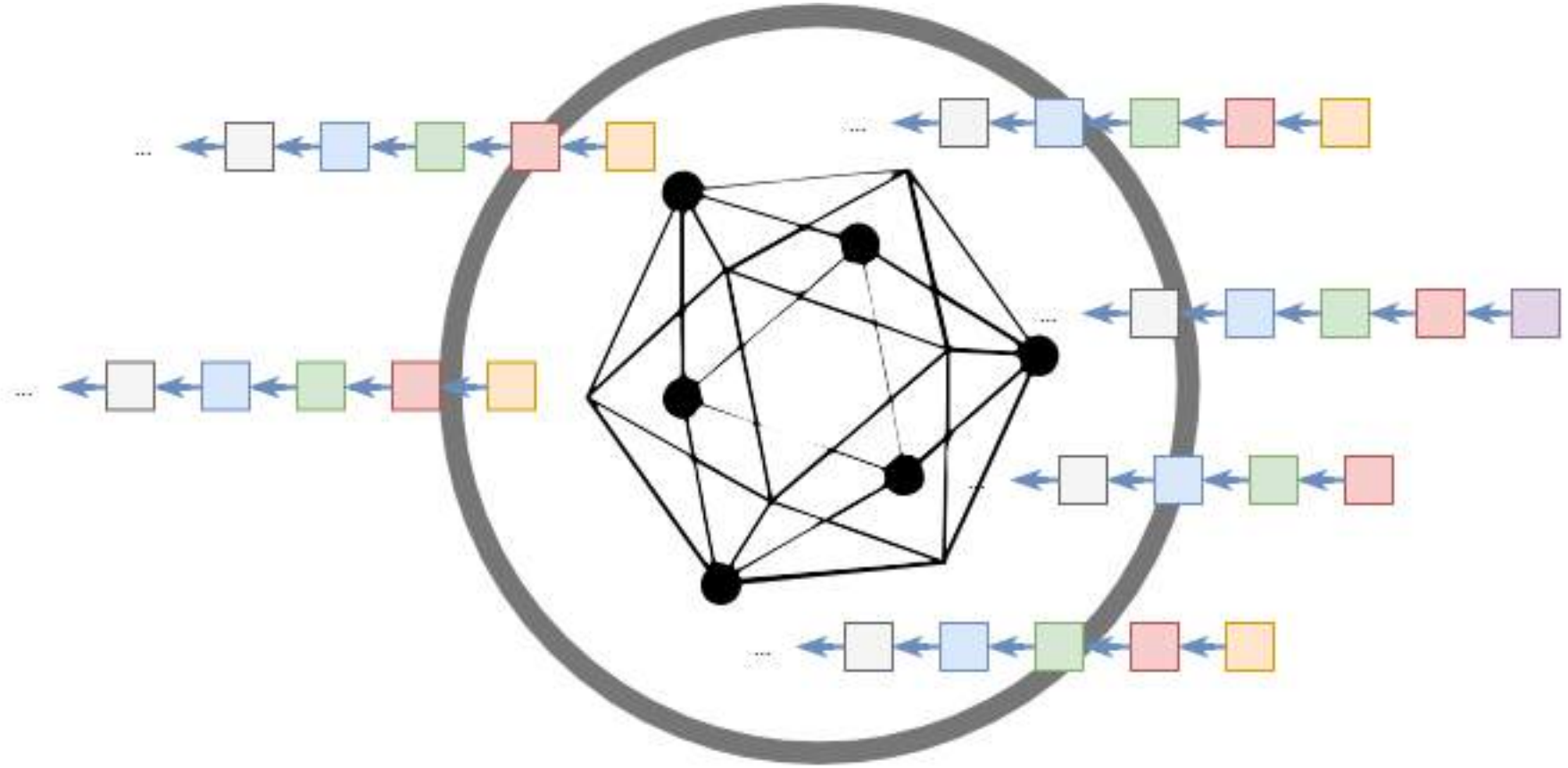
Where are Smart Contracts executed?

First on the mining nodes.
Then, potentially, on every node!

Only absolutely needed instructions should be in the code!



Distributed nature



Smart contracts are pieces of code (not for free)

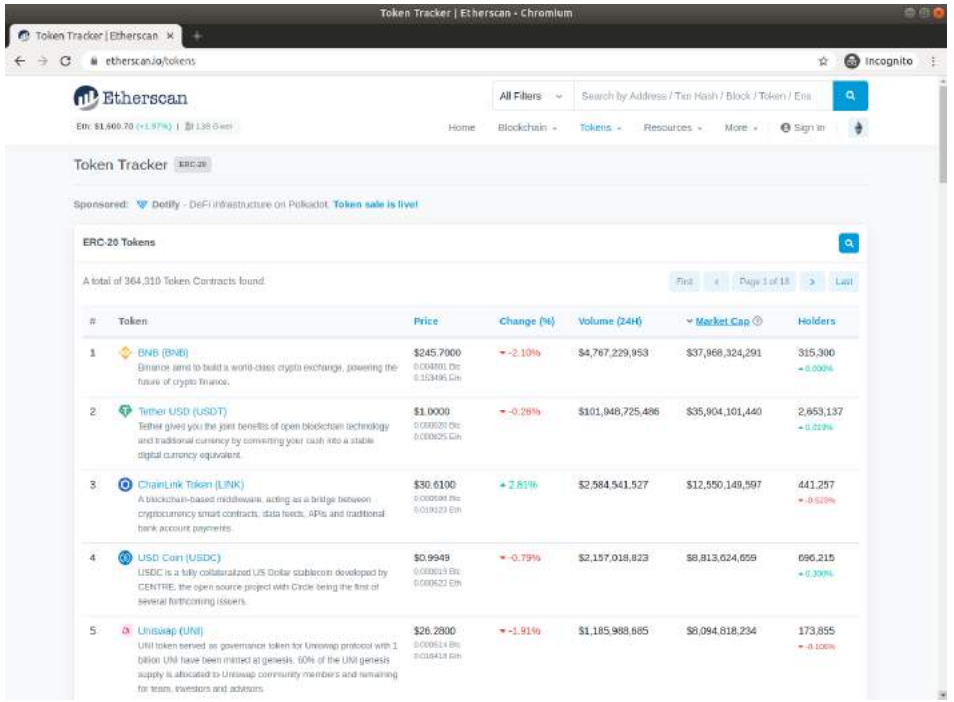
```solidity
pragma solidity ^0.4.0;

contract HelloToken {
    address public minter;
    mapping (address => uint) public balance;
    uint public constant PRICE = 2 finney;

    constructor() public {
        minter = msg.sender;
    }

    function mint() public payable {
        require(msg.value >= PRICE, "Not enough value for a token!");
        balance[msg.sender] += msg.value / 2 finney;
    }

    function transfer(uint amount, address to) public {
        require(balance[msg.sender] >= amount, "Not enough tokens!");
        balance[msg.sender] -= amount;
        balance[to] += amount;
    }

    function terminate() public {
        require(msg.sender == minter, "You cannot terminate the contract!");
        selfdestruct(minter);
    }
}
```

Name	Value	Description*
G_{zero}	0	Nothing paid for operations of the set W_{zero}.
G_{base}	2	Amount of gas to pay for operations of the set W_{base}.
$G_{verylow}$	3	Amount of gas to pay for operations of the set $W_{verylow}$.
G_{low}	5	Amount of gas to pay for operations of the set W_{low}.
G_{mid}	8	Amount of gas to pay for operations of the set W_{mid}.
G_{high}	10	Amount of gas to pay for operations of the set W_{high}.
$G_{extcode}$	700	Amount of gas to pay for operations of the set $W_{extcode}$.
$G_{balance}$	400	Amount of gas to pay for a BALANCE operation.
G_{sload}	200	Paid for a SLOAD operation.
$G_{jumpdest}$	1	Paid for a JUMPDEST operation.
G_{sset}	20000	Paid for an SSTORE operation when the storage value is set to non-zero from zero.
G_{sreset}	5000	Paid for an SSTORE operation when the storage value's zeroness remains unchanged or is set to zero.
R_{sclear}	15000	Refund given (added into refund counter) when the storage value is set to zero from non-zero.
$R_{selfdestruct}$	24000	Refund given (added into refund counter) for self-destructing an account.
$G_{selfdestruct}$	5000	Amount of gas to pay for a SELFDESTRUCT operation.
G_{create}	32000	Paid for a CREATE operation.
$G_{codedeposit}$	200	Paid per byte for a CREATE operation to succeed in placing code into state.
G_{call}	700	Paid for a CALL operation.
$G_{callvalue}$	9000	Paid for a non-zero value transfer as part of the CALL operation.
$G_{callstipend}$	2300	A stipend for the called contract subtracted from $G_{callvalue}$ for a non-zero value transfer.
$G_{newaccount}$	25000	Paid for a CALL or SELFDESTRUCT operation which creates an account.
G_{exp}	10	Partial payment for an EXP operation.
$G_{expbyte}$	50	Partial payment when multiplied by $\lceil \log_{256}(exponent) \rceil$ for the EXP operation.
G_{memory}	3	Paid for every additional word when expanding memory.
$G_{txcreate}$	32000	Paid by all contract-creating transactions after the *Homestead* transition.
$G_{txdatazero}$	4	Paid for every zero byte of data or code for a transaction.
$G_{txdatanonzero}$	68	Paid for every non-zero byte of data or code for a transaction.
$G_{transaction}$	21000	Paid for every transaction.
G_{log}	375	Partial payment for a LOG operation.
$G_{logdata}$	8	Paid for each byte in a LOG operation's data.
$G_{logtopic}$	375	Paid for each topic of a LOG operation.
G_{sha3}	30	Paid for each SHA3 operation.
$G_{sha3word}$	6	Paid for each word (rounded up) for input data to a SHA3 operation.
G_{copy}	3	Partial payment for *COPY operations, multiplied by words copied, rounded up.
$G_{blockhash}$	20	Payment for BLOCKHASH operation.
$G_{quaddivisor}$	100	The quadratic coefficient of the input sizes of the exponentiation-over-modulo precompiled contract.

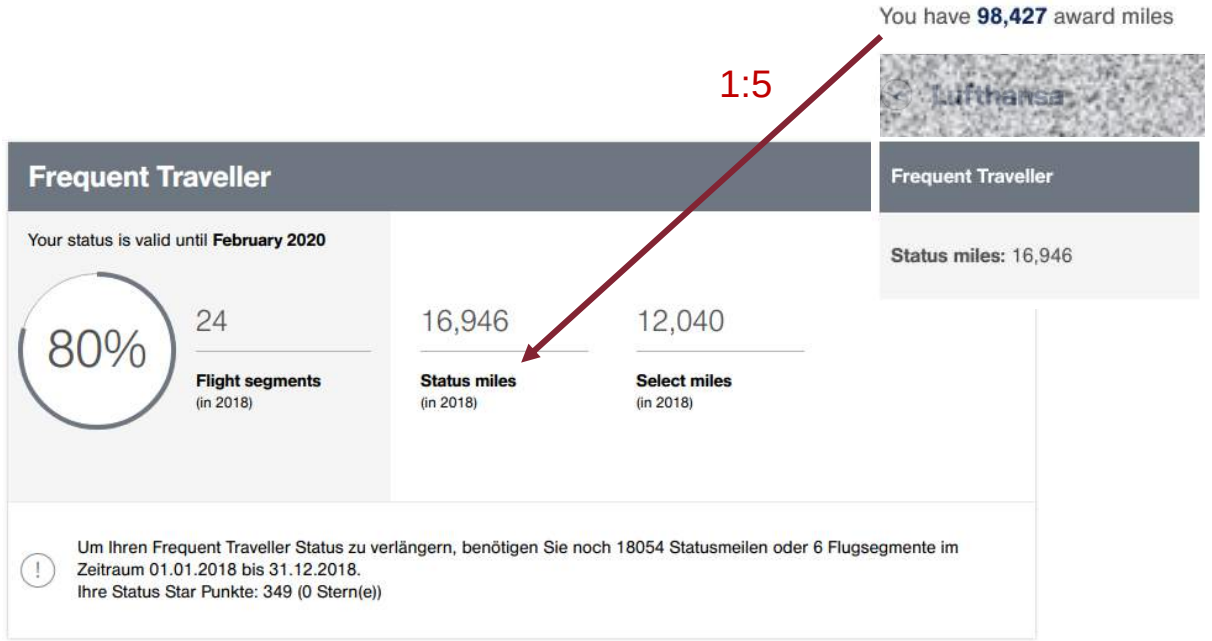
Wood, "Ethereum: A secure decentralised generalised transaction ledger" (2014), https://ethereum.github.com/yellowpaper/paper.pdf

Smart contracts and tokens

```
pragma solidity ^0.4.0;

contract HelloToken {
    address public minter;
    mapping (address => uint) public balance;
    uint public constant PRICE = 2 finney;

    constructor() public {
        minter = msg.sender;
    }

    function mint() public payable {
        require(msg.value >= PRICE, "Not enough value for a token!");
        balance[msg.sender] += msg.value / 2 finney;
    }

    function transfer(uint amount, address to) public {
        require(balance[msg.sender] >= amount, "Not enough tokens!");
        balance[msg.sender] -= amount;
        balance[to] += amount;
    }

    function terminate() public {
        require(msg.sender == minter, "You cannot terminate the contract!");
        selfdestruct(minter);
    }
}

```

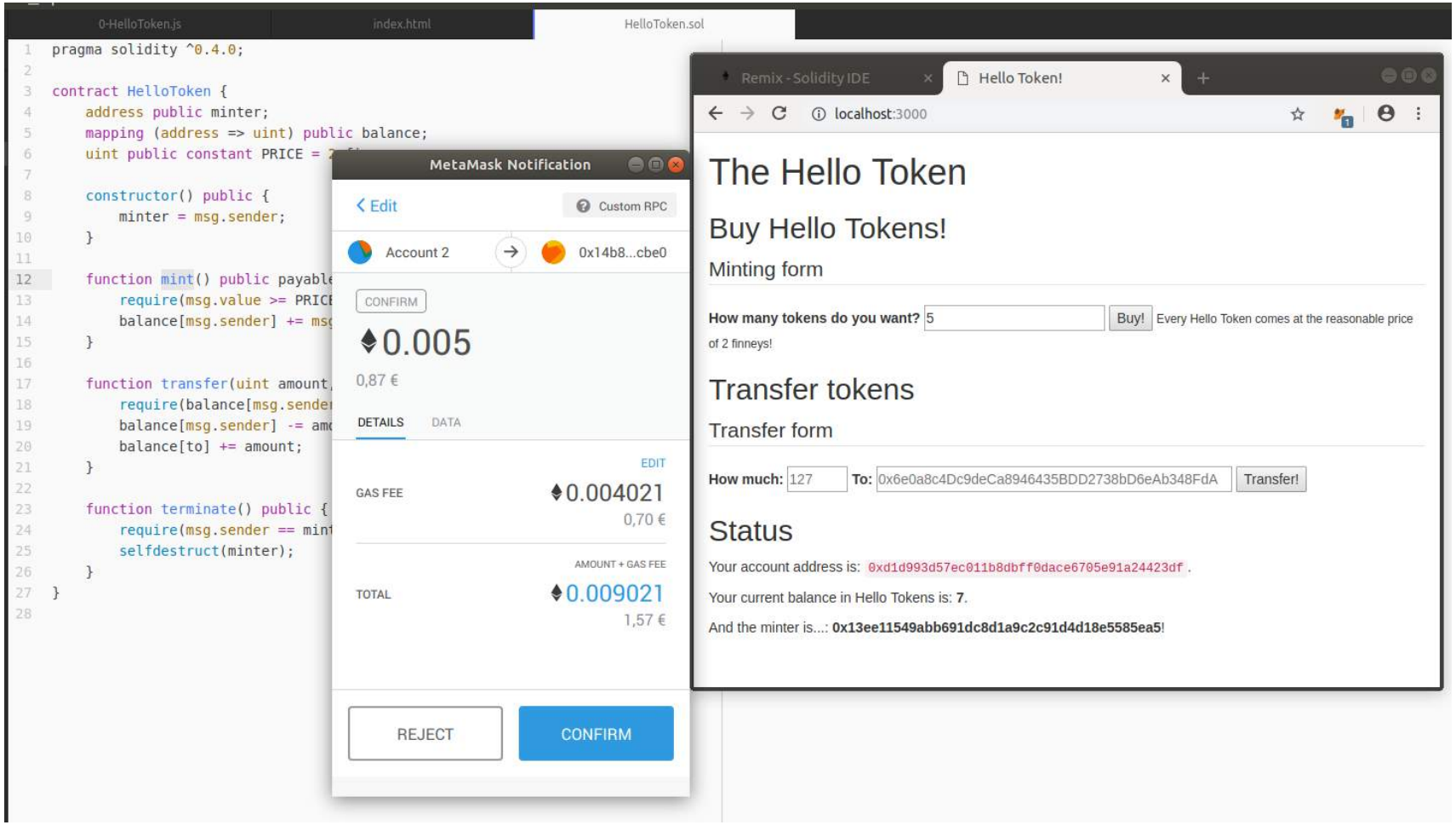


Tokens are not cryptofuel nor anything conceptually new, after all!



Period	01.01.2017 to today
Number of flights	66
Flown distance	81,273 km or 50,501 miles
Flight time	5 d 21 h 0 min

Your brand new token in 5 minutes or less



Tokens

Token: A new asset class

- Multiple functions:
  - Currency
    - large-entity-backed medium of exchange for goods and services
  - Commodity
    - basic good tradeable or exchangeable with other goods of the same type
  - Utility
    - Satisfaction quantifier for an economic good or service
    - “Utility token” is even on the Merriam-Wesbter dictionary: https://www.merriam-webster.com/dictionary/utility%20token
  - Security
    - Financial instrument that guarantee ownership, credit, or decision power

- Different nature:
  - Fungible
    - Individual units can be mutually substituted
  - Non-fungible
    - Units are unique and not interchangeable

Private | public / Permissioned | permissionless

Transactability / visibility

Consensus

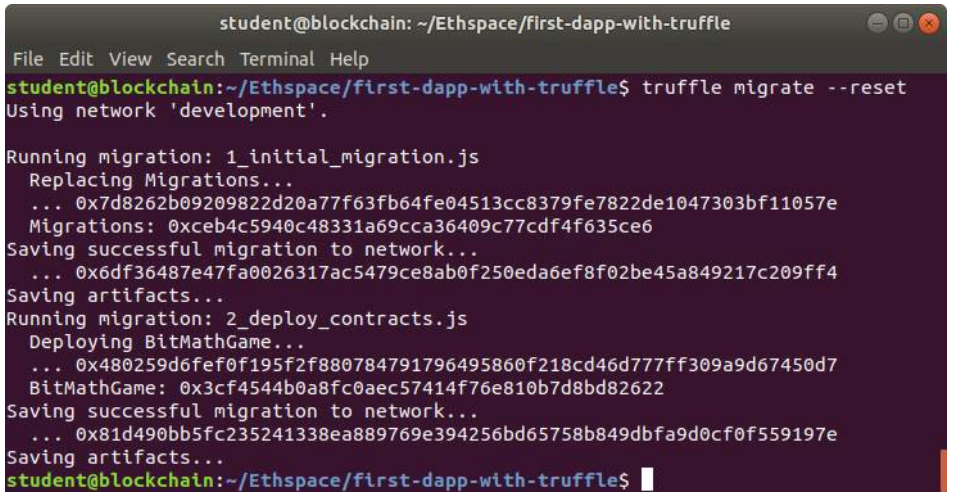
Anderberg et al. “Blockchain Now And Tomorrow” (2019) http://dx.doi.org/10.2760/901029

Private | public / Permissioned | permissionless

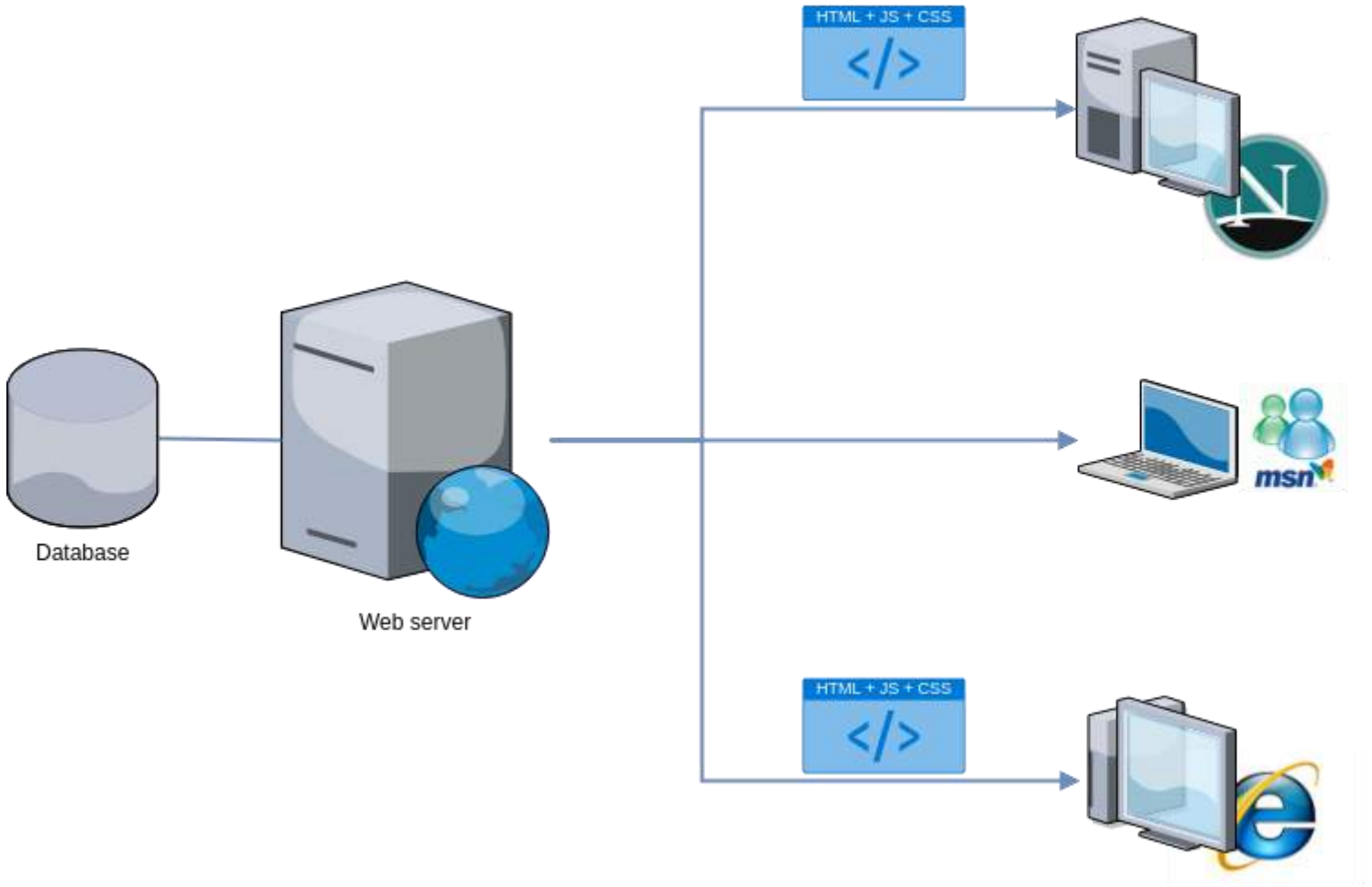
		Transactability / visibility	
		Private	Public
Consensus	Permissionless	**Selected** nodes can transact and view, **all** nodes can participate in consensus	**Every** node can transact and view, participate in consensus
	Permissioned	**Selected** nodes can transact and view, or participate in consensus	**Every** node can transact and view, **selected** nodes participate in consensus

Anderberg et al. “Blockchain Now And Tomorrow” (2019) http://dx.doi.org/10.2760/901029

The Blockchain and the Internet



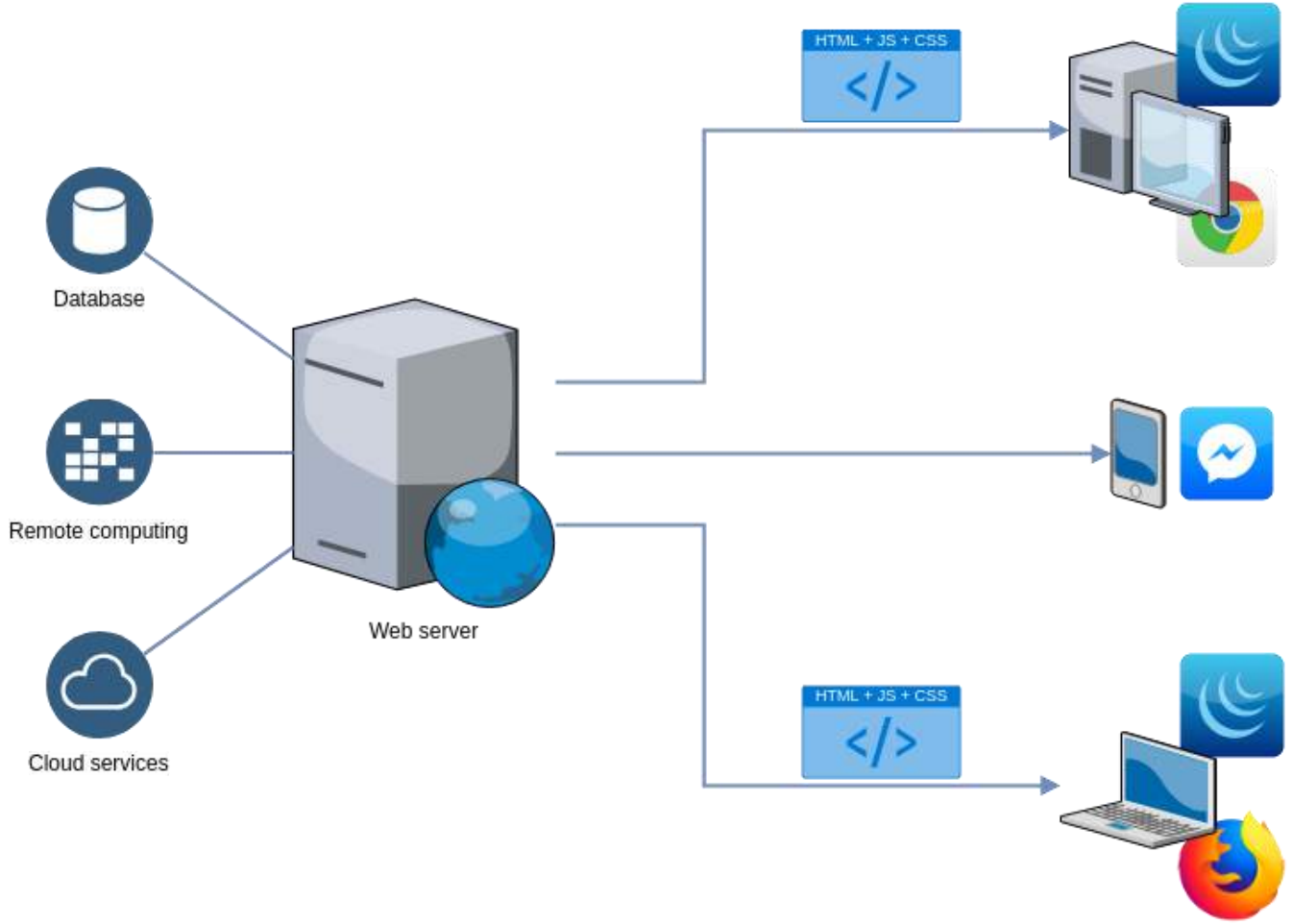
Web 1.0



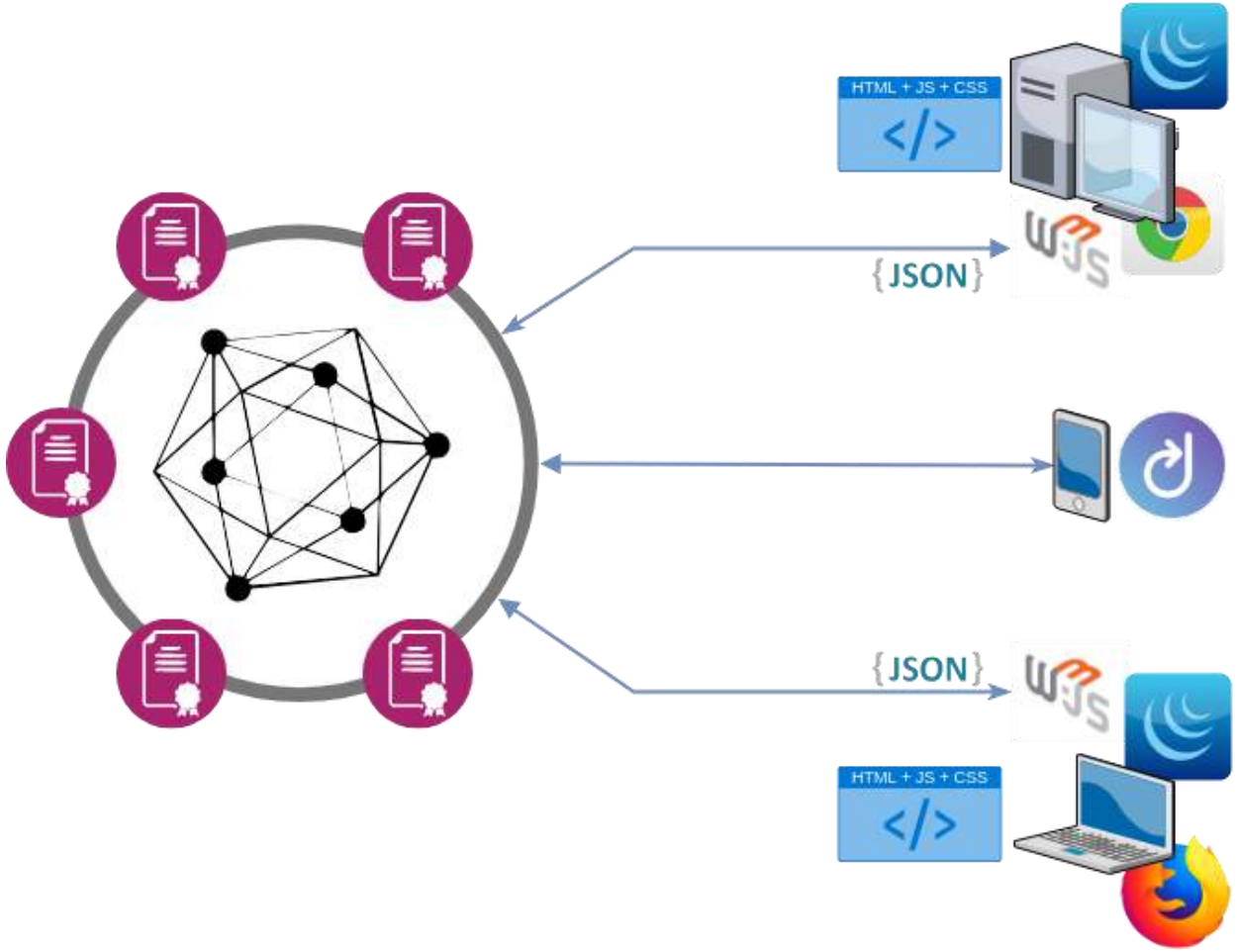
Back end

Front end

Web 2.0


HTML + JS + CSS
Database
Remote computing
Cloud services
Web server
HTML + JS + CSS

Web 3.0
Decentralised applications (DApps)



Wrap-up

- Transactions → Transfer of assets (and code invocation)
- Signatures → Authentication
- Ledger → Transaction ordering
- Distributed architecture → Data persistency
- Hashing → Robustness
- Proof-of-[…] → Publishing rights
- Consensus → Eventual consistency
- Smart contracts → Programmability (and tokens / app coins)

Blockchain and Distributed Ledger Technologies

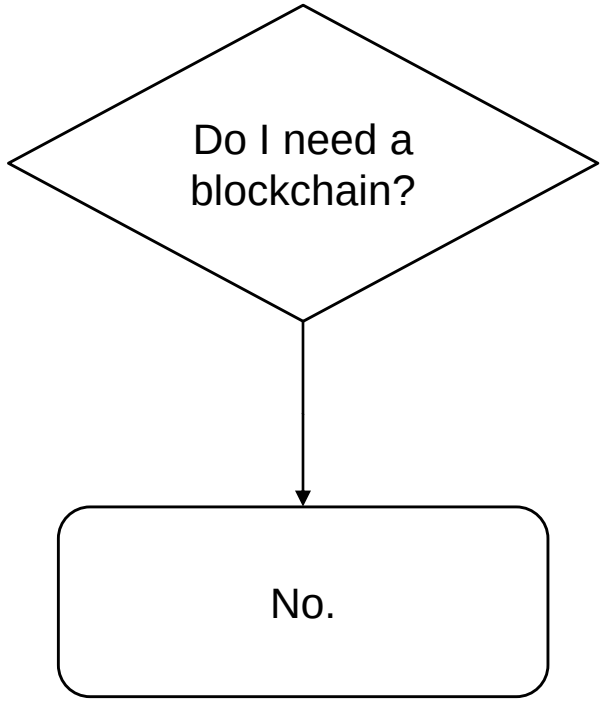
M.Sc. course at the Sapienza University of Rome

Starting in September 2022 (a.y. 2022-23, 1st semester)

Do I need the blockchain then?

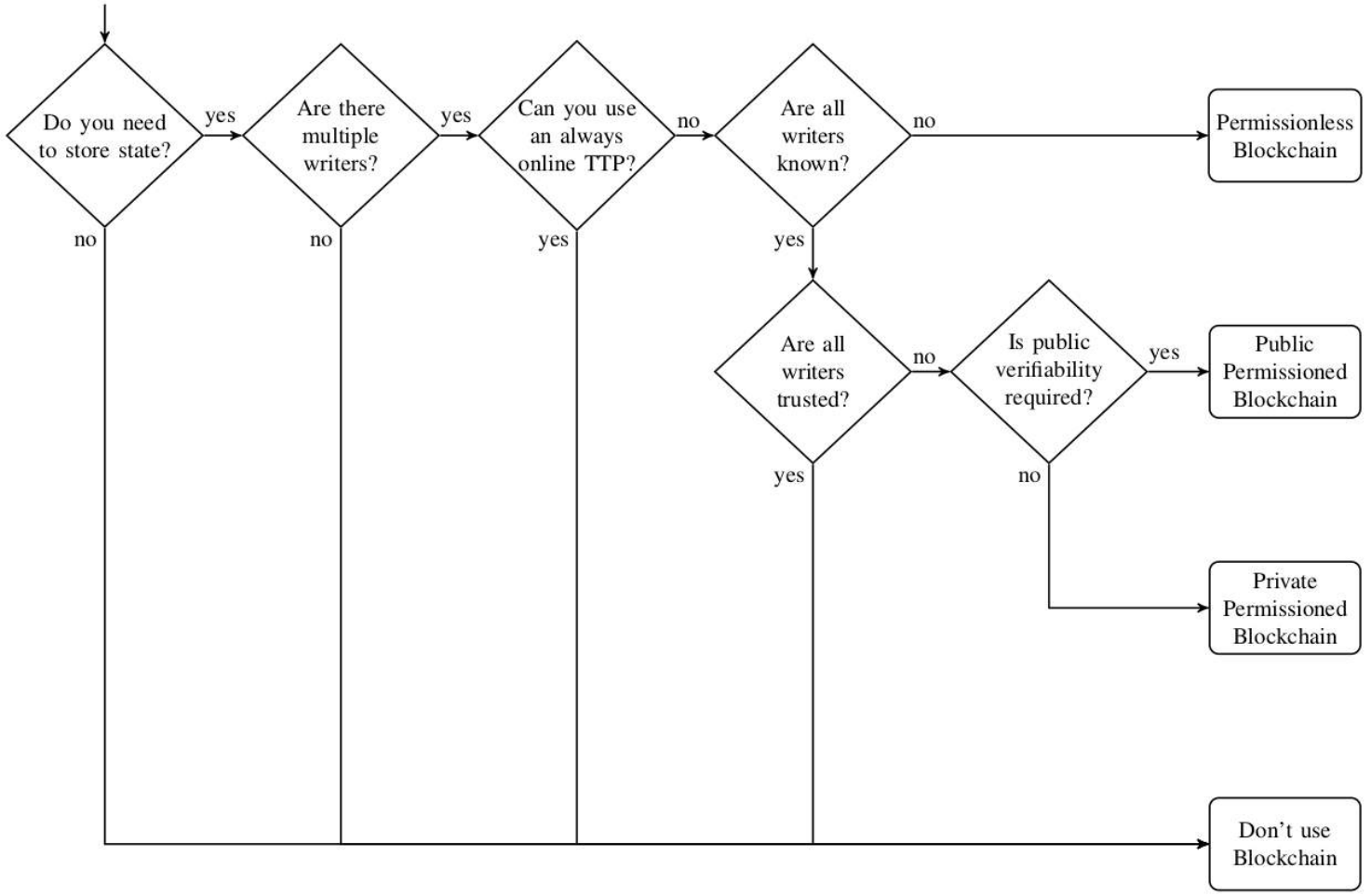
Interlude

Do I need a blockchain? (Birch model, joke)



https://medium.com/@sbmeunier/when-do-you-need-blockchain-decision-models-a5c40e7c9ba1

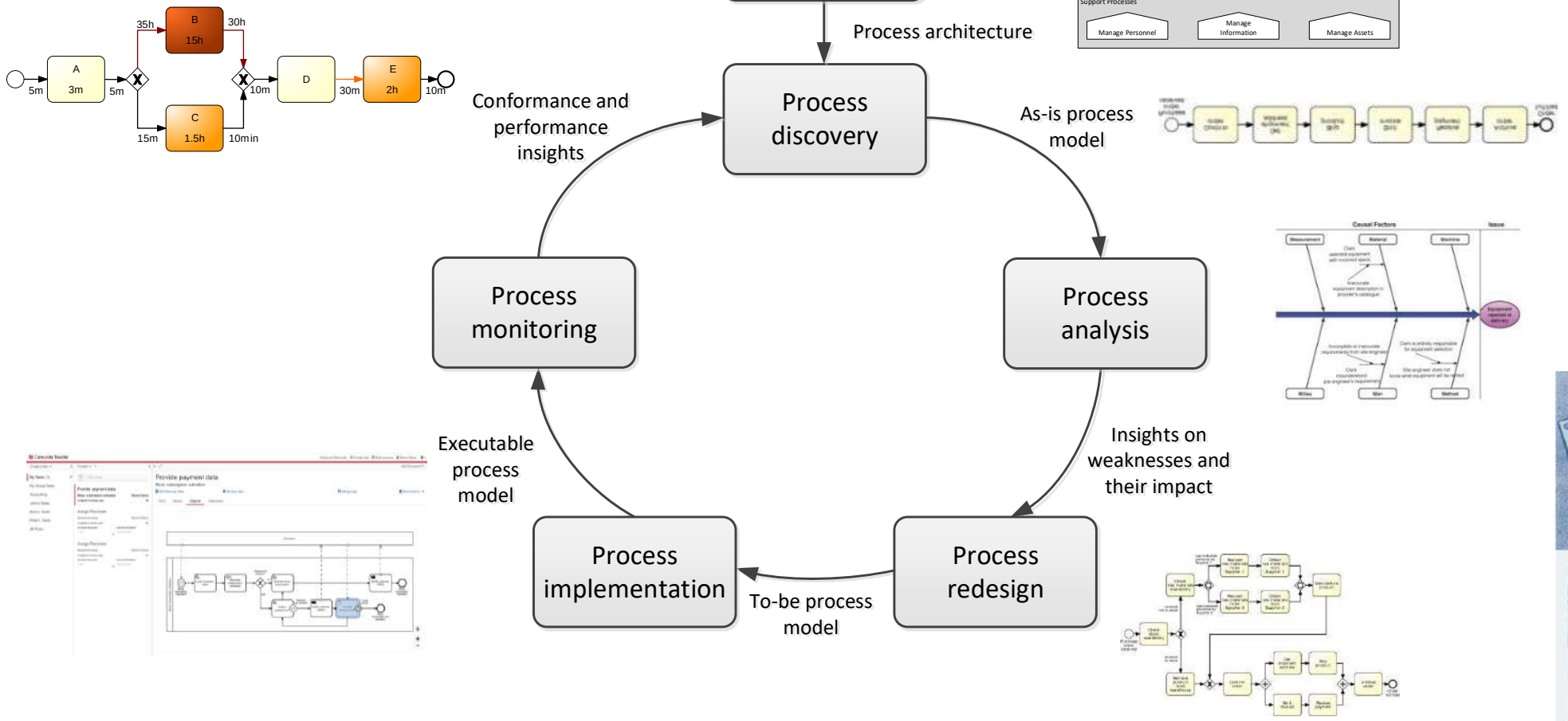
Do I need a blockchain? (Wüst & Gervais)

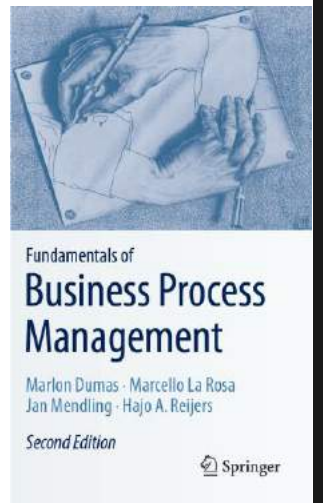


Wüst, K. & Gervais, A. Do you Need a Blockchain? CVCBT, IEEE, 2018

Blockchain as a process execution infrastructure

Process Science and Business Process Management





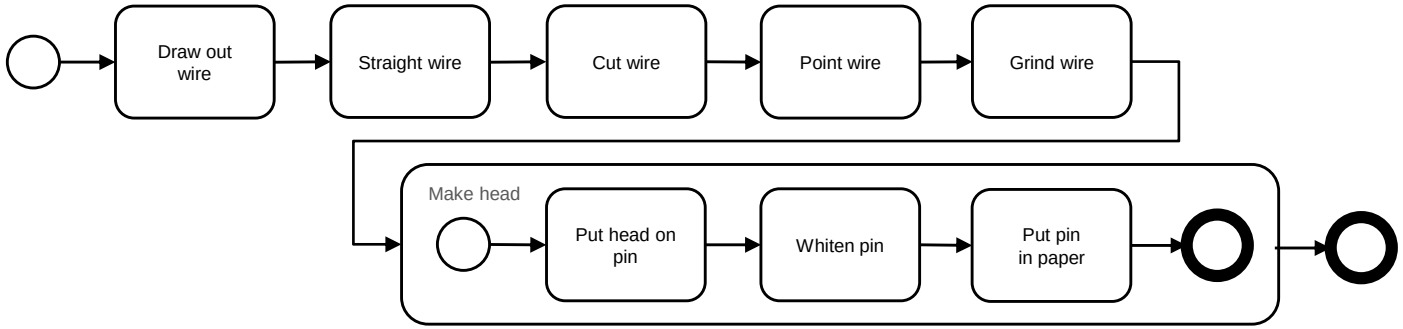
Dumas, La Rosa, Mendling, Reijers: Fundamentals of Business Process Management, Second Edition. Springer, 2018.

Processes and division of labour

The trade of a pin-maker: But in the way in which this business is now carried on, it is divided into a number of branches:

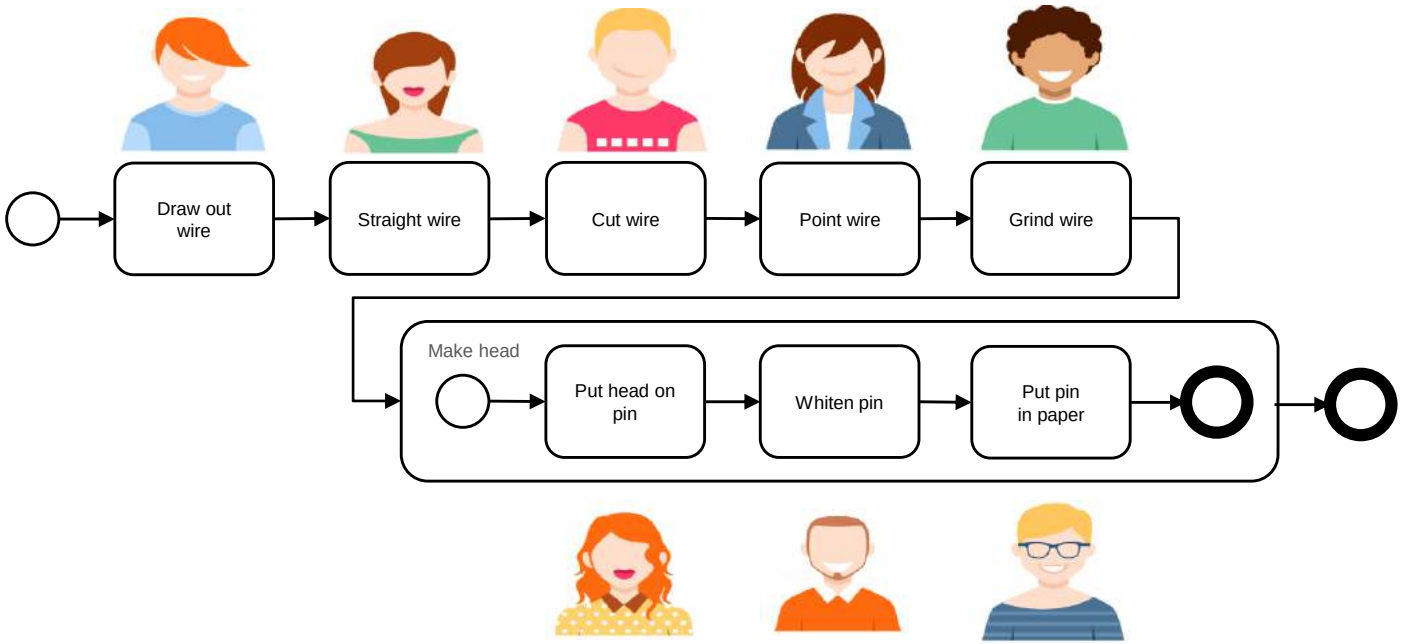
- One man draws out the wire;
- another straights it;
- a third cuts it;
- a fourth points it;
- a fifth grinds it at the top for receiving the head;
- to make the head requires three operations;
  - to put it on is a peculiar business;
  - to whiten the pins is another;
  - to put them into the paper; …

[…] making a pin is, in this manner, divided into about 18 distinct operations.

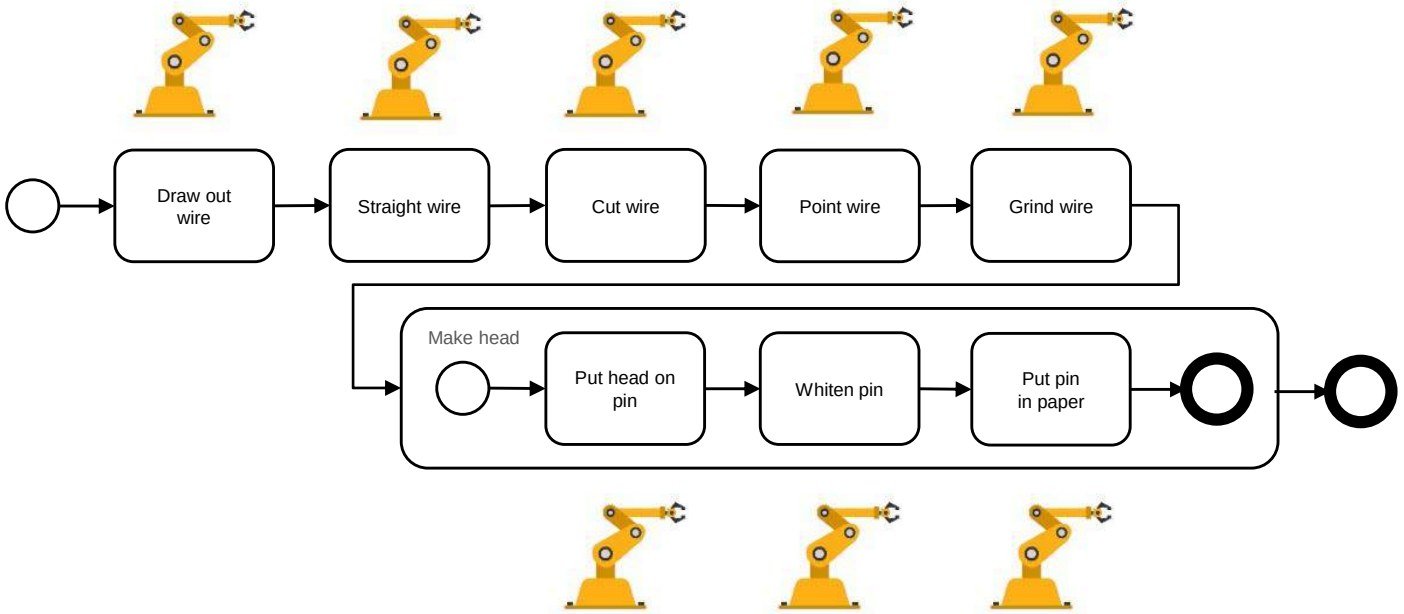


[1] Smith 1776 [2] Dumas, La Rosa, Mendling, Reijers 2018

Division of labour



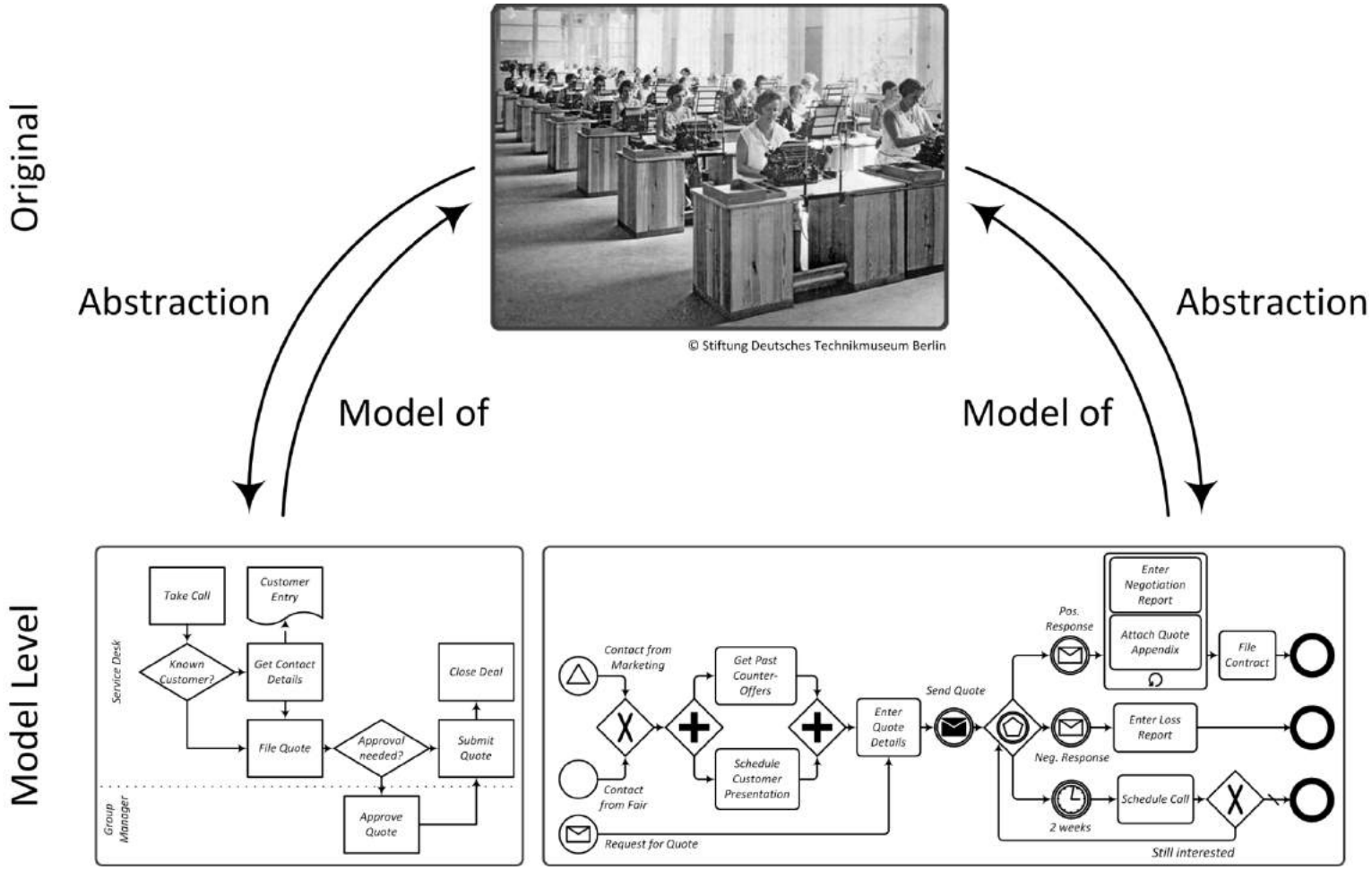
Division of labour → Automation



Systems like to report on their job (logging)

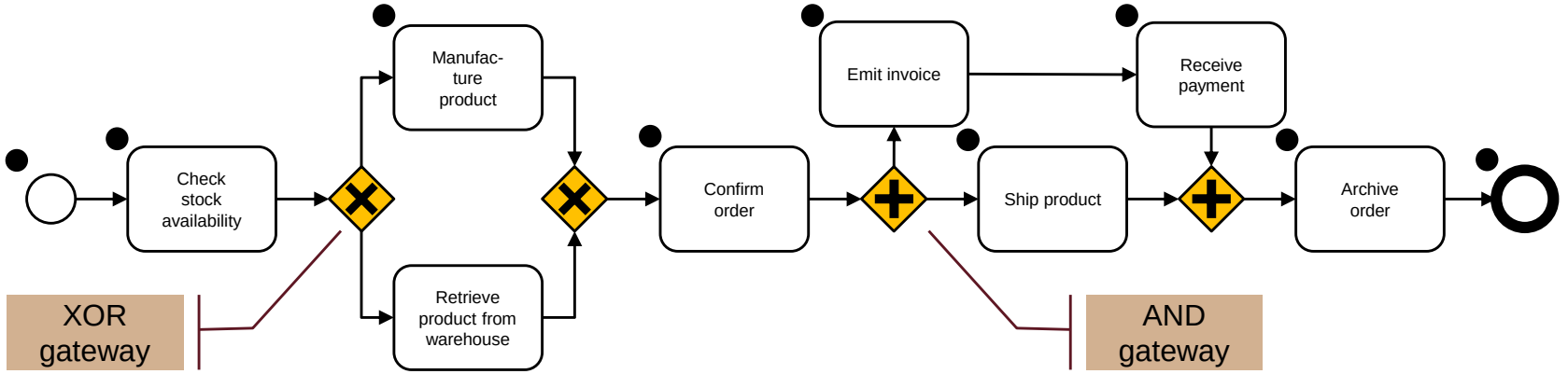
- [2019-02-18T12:30:00-02:00] 0xACDC0801 executes Draw Out Wire on Item 0xAA01
- [2019-02-18T12:30:10-02:00] 0xACDC0802 executes Straight Wire on Item 0xAA01
- [2019-02-18T12:30:20-02:00] 0xACDC0803 executes Cut Wire on Item 0xAA01
- [2019-02-18T12:30:30-02:00] 0xACDC0801 executes Draw Out Wire on Item 0xAA02
- [2019-02-18T12:30:40-02:00] 0xACDC0802 executes Straight Wire on Item 0xAA02
- [2019-02-18T12:30:50-02:00] 0xACDC0804 executes Point Wire on Item 0xAA01
- [2019-02-18T12:31:00-02:00] 0xACDC0801 executes Draw Out Wire on Item 0xAA03
- …

Process models

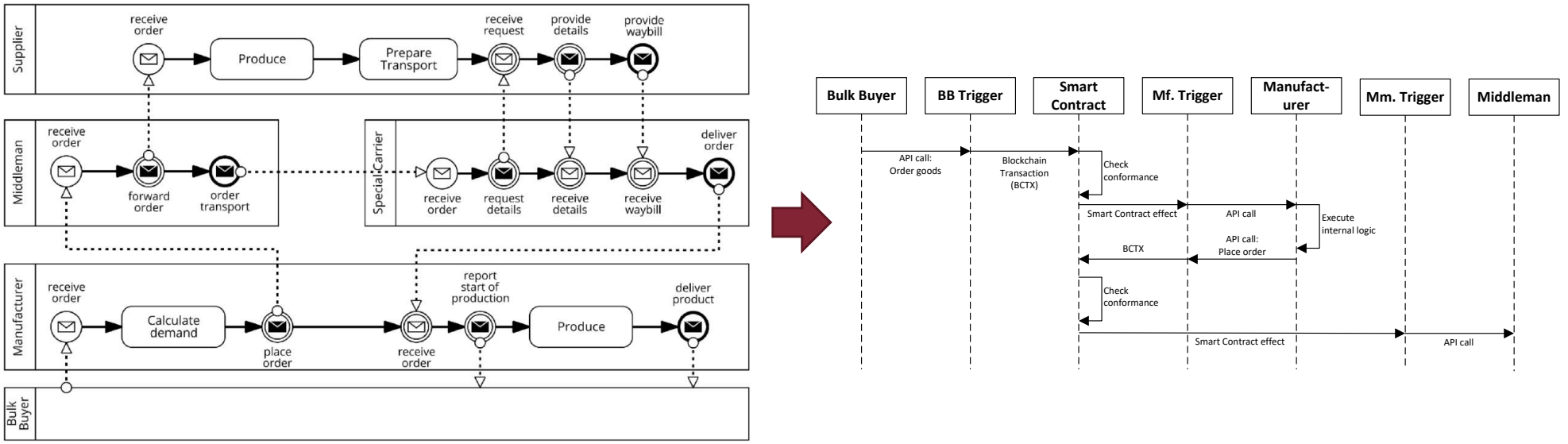


Slide courtesy of Jan Mendling

Business Process Modelling and Notation (BPMN)

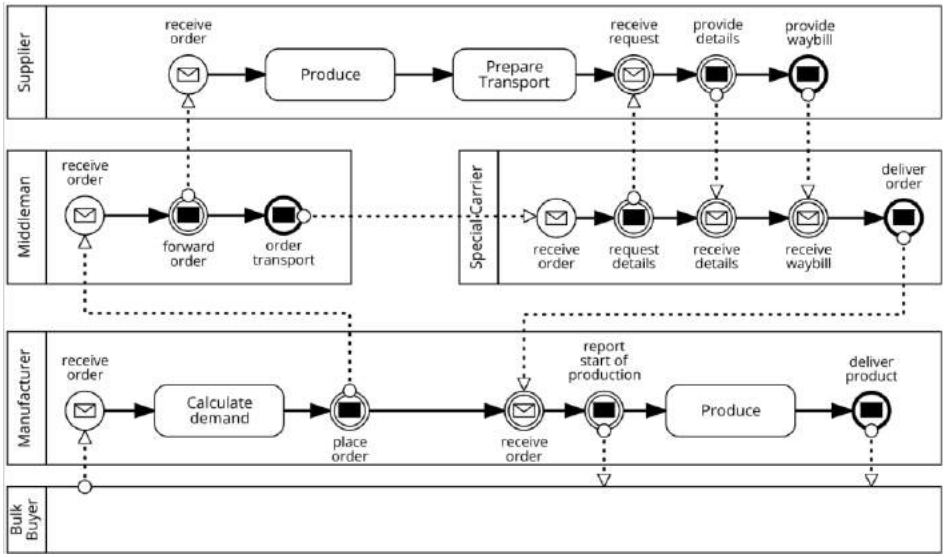


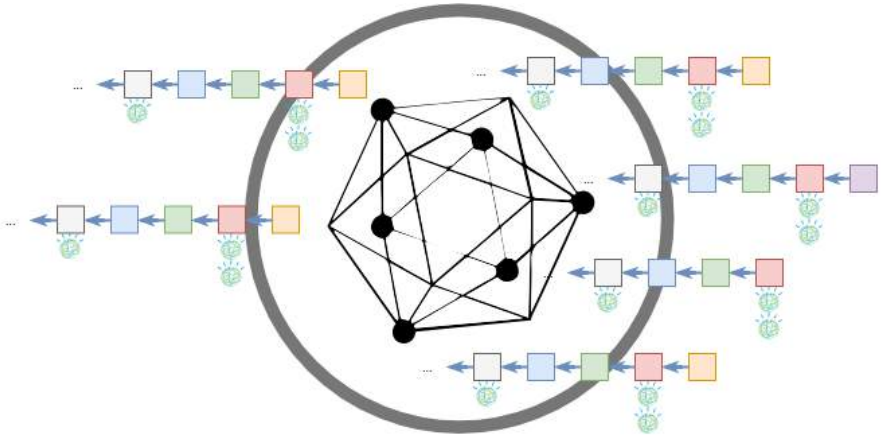
Smart contracts can execute processes

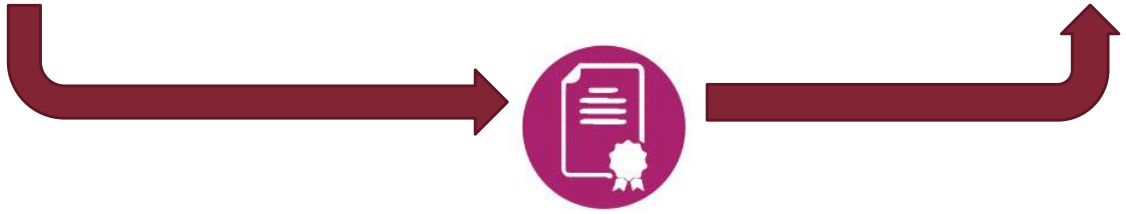


Weber et al.: Untrusted business process monitoring and execution using blockchain. In: Proc. of BPM. Springer, 2016.

Executing inter-organisational processes on the Blockchain: A model-driven approach





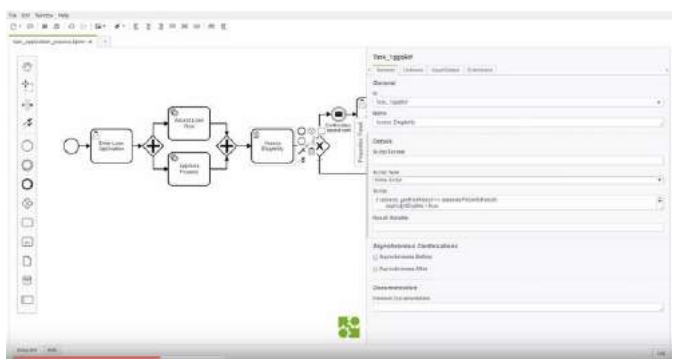


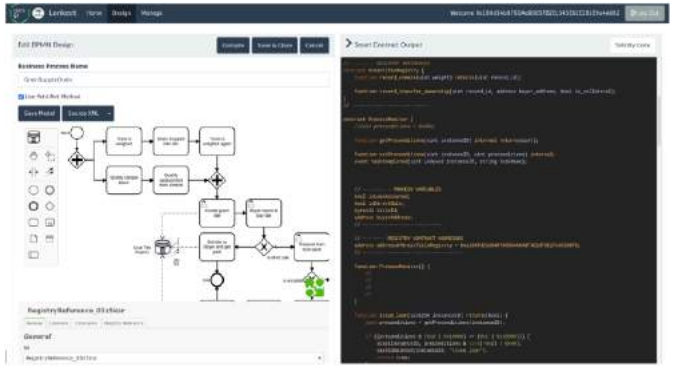
D.C, Cecconi, Dumas et al. Blockchain support for collaborative business processes. Informatik Spektrum pp. 1–9 (May 2019)

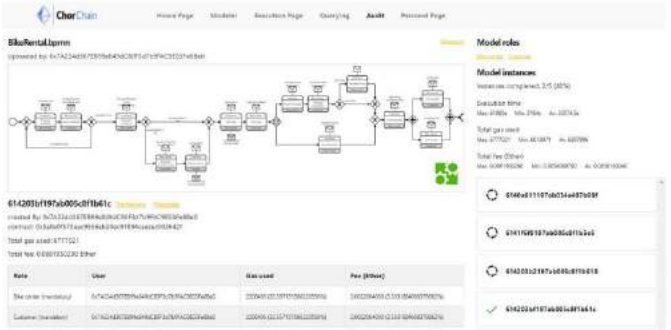
Executing inter-organisational processes on the Blockchain: A model-driven approach

López-Pintado, García-Bañuelos, Dumas, Weber. **Caterpillar**: A blockchain-based business process management system. In: BPM Demos. CEUR.ws, 2017.
Tran, Lu, Weber. **Lorikeet**: A Model-Driven Engineering Tool for Blockchain-Based Business Process Execution and Asset. In: BPM Demos. CEUR.ws, 2018.
Corradini, Marcelletti, Morichetta, Polini, Re, Tiezzi: Engineering Trustable and Auditable Choreography-based Systems Using Blockchain. ACM TMIS 13(3), 2022.

Caterpillar | Lorikeet | ChorChain





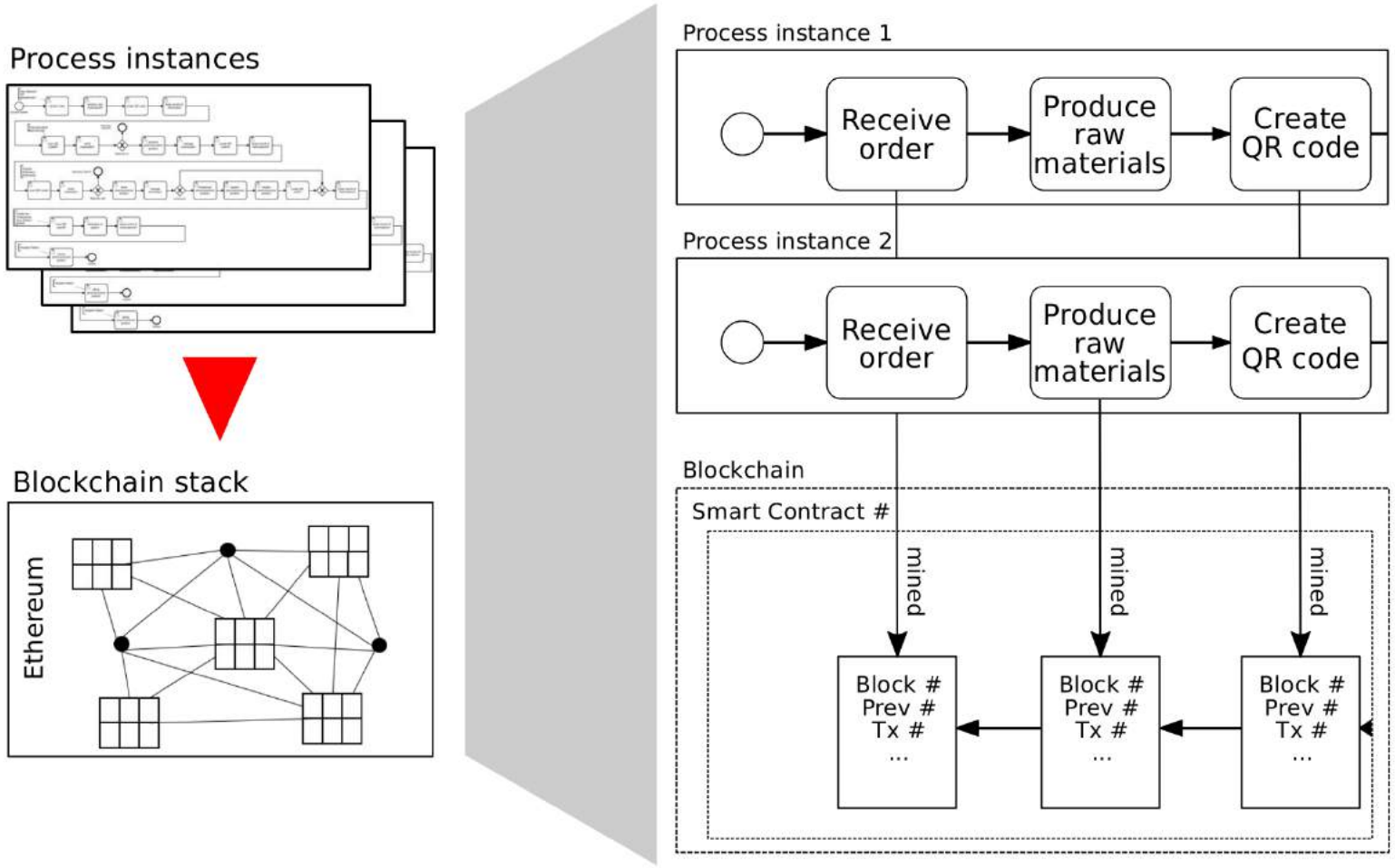


D.C, Cecconi, Dumas et al. Blockchain support for collaborative business processes. Informatik Spektrum pp. 1–9 (May 2019)

Rationale

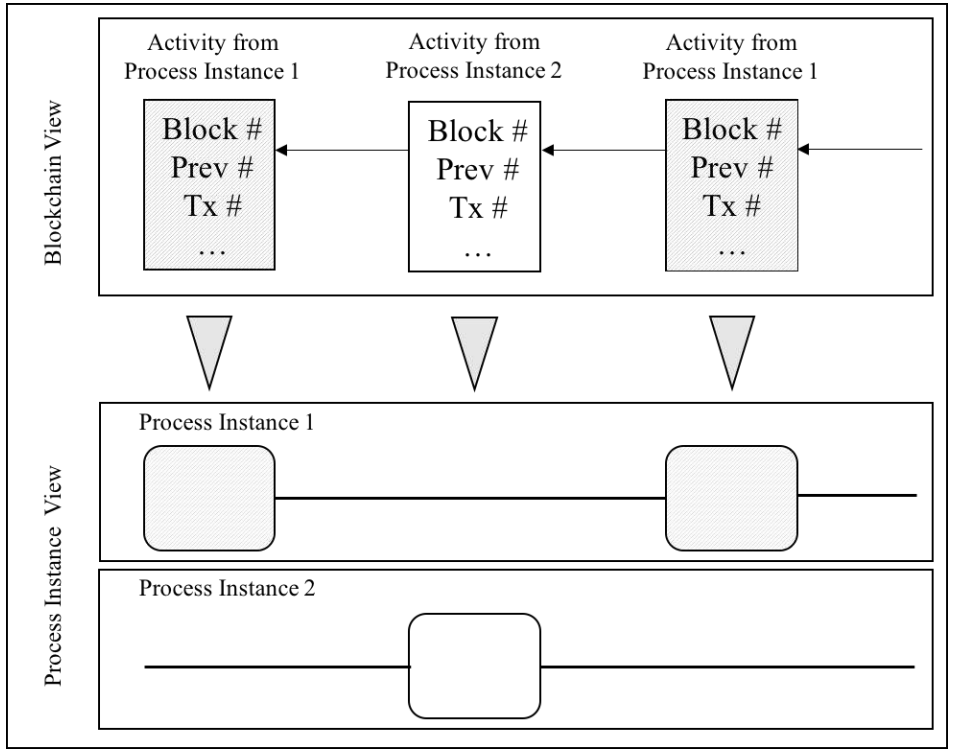
Blockchain-Based Traceability of Inter-organisational Business Processes

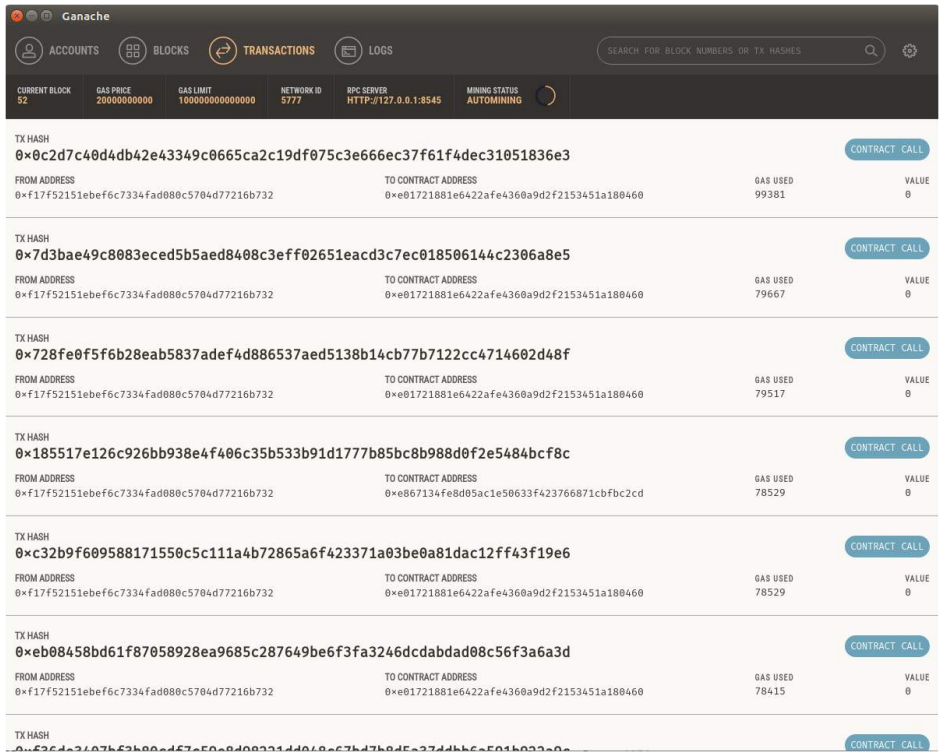
Claudio Di Ciccio, Alessio Cecconi, Jan Mendling, Dominik Felix, Dominik Haas, Daniel Lilek, Florian Riel, Andreas Rumpl, and Philipp Uhlig



D.C. et al. Blockchain-Based Traceability of Inter-organisational Business Processes. In: BMSD. Springer, 2018.

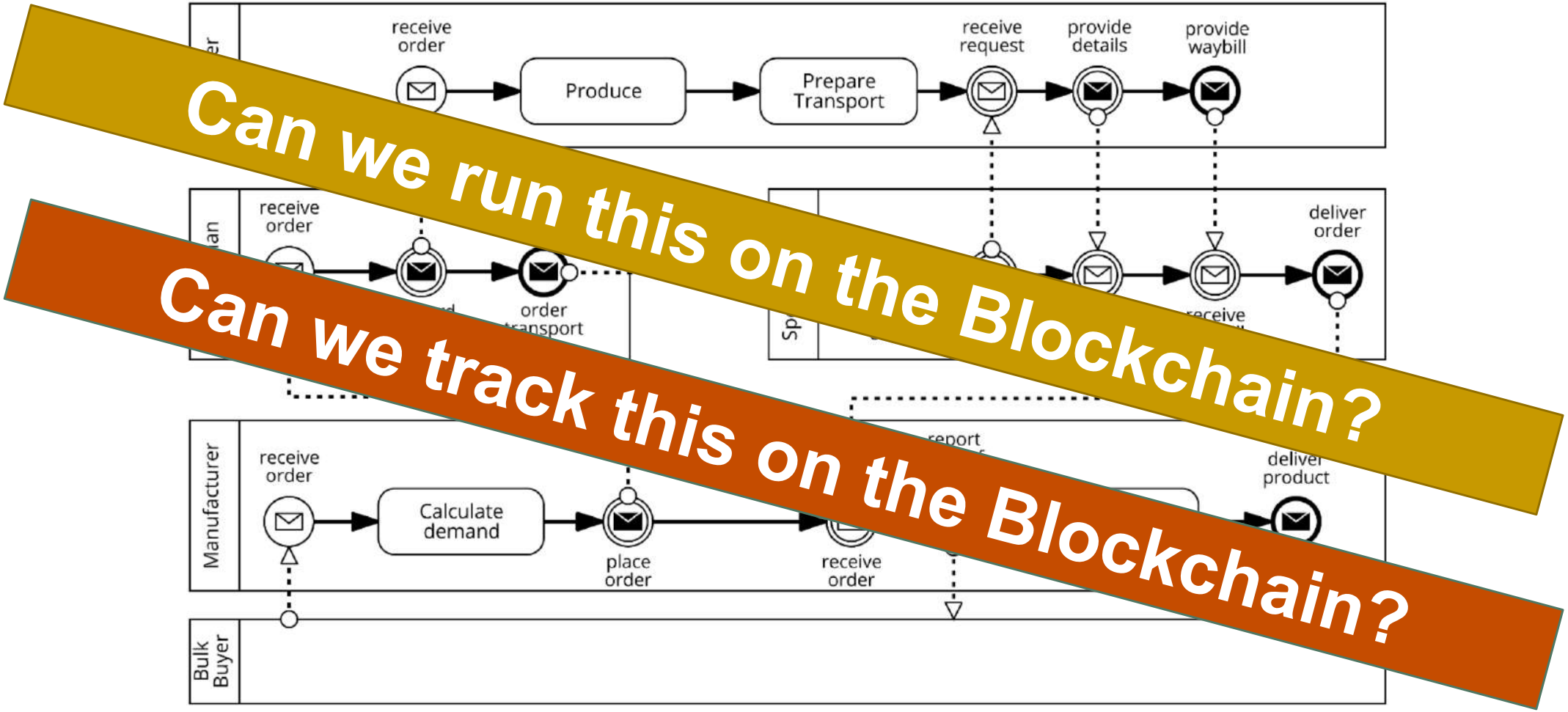
Rationale



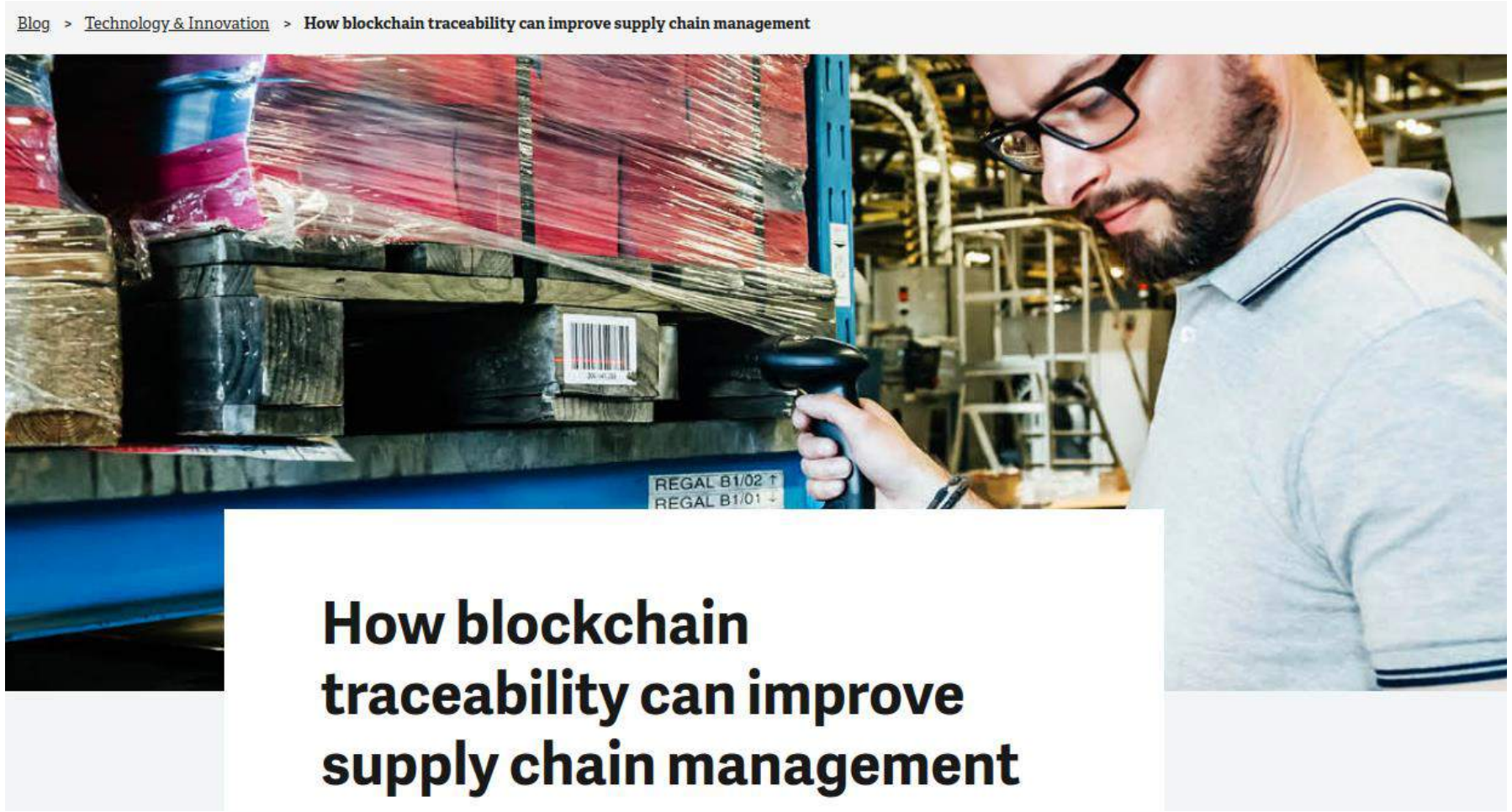


D.C. et al. Blockchain-Based Traceability of Inter-organisational Business Processes. In: BMSD. Springer, 2018.

A multi-organisational process



Traceability



https://www.sage.com/en-gb/blog/blockchain-traceability-supply-chain/ Accessed: 19/02/2020

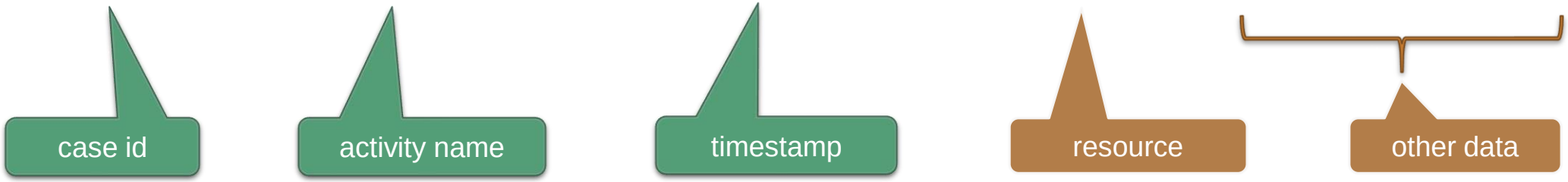
Mining for processes

Discovery, conformance, and enhancement of processes based on data

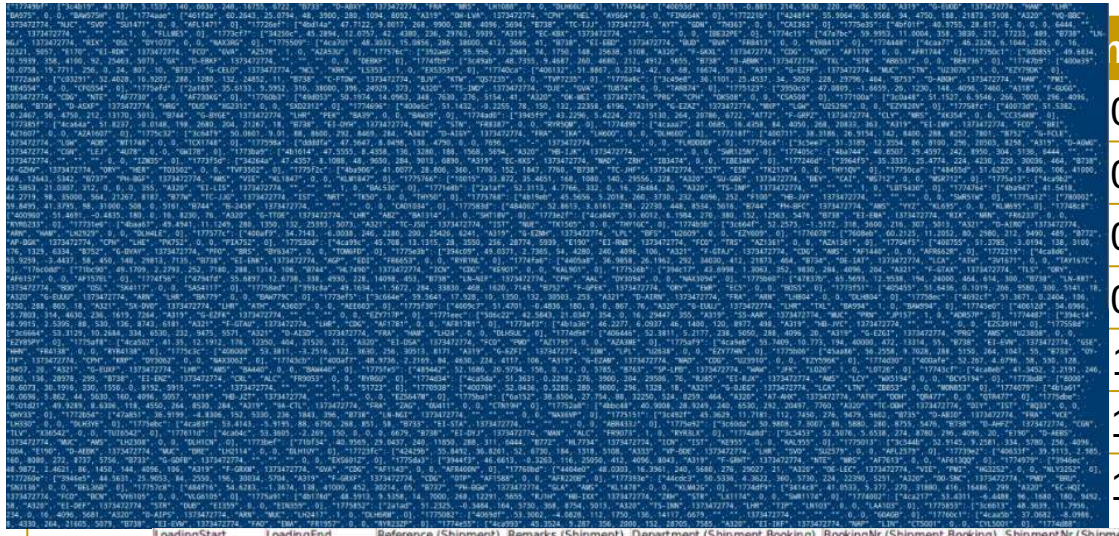
On the right: https://hbr.org/2019/04/what-process-mining-is-and-why-companies-should-do-it

An event log (tabular form)

order n.	activity	timestamp	user	product	qty.
9901	register order	22-1-2014@09.15	Sara Jones	iPhone5S	1
9902	register order	22-1-2014@09.18	Sara Jones	iPhone5S	2
9903	register order	22-1-2014@09.27	Sara Jones	iPhone4S	1
9901	check stock	22-1-2014@09.49	Pete Scott	iPhone5S	1
9901	ship order	22-1-2014@10.11	Sue Fox	iPhone5S	1
9903	check stock	22-1-2014@10.34	Pete Scott	iPhone4S	1
9901	handle payment	22-1-2014@10.41	Carol Hope	iPhone5S	1
9902	check stock	22-1-2014@10.57	Pete Scott	iPhone5S	2
9902	cancel order	22-1-2014@11.08	Carol Hope	iPhone5S	2
…	…	…	…	…	…



Event logs v. data at hand

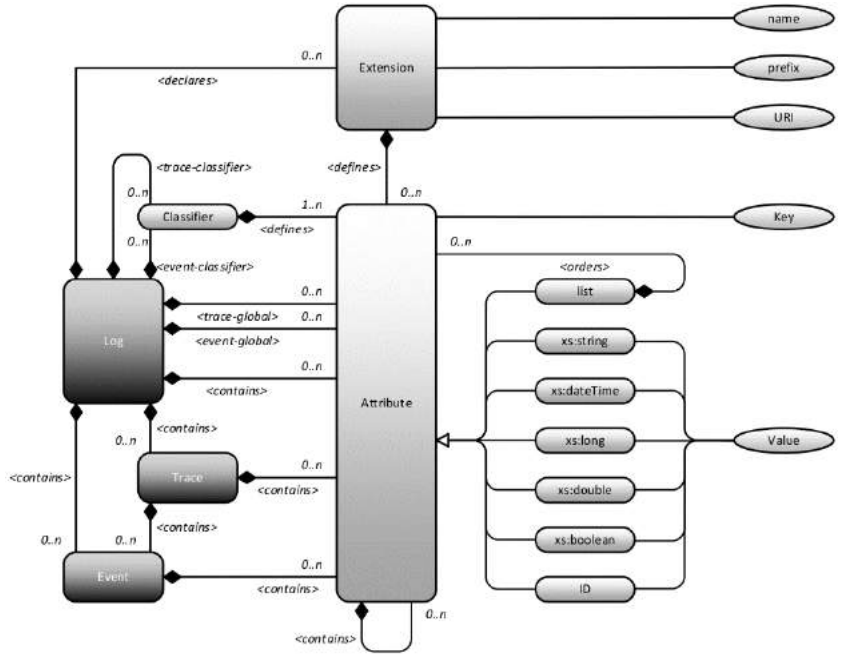


…mp	qty.
09.15	1
09.18	2
09.27	1
09.49	1
10.11	1
10.34	1
10.41	1
9	2
9	2
	…

LoadingStart	LoadingEnd	Reference (Shipment)	Remarks (Shipment)	Department (Shipment,Booking)	BookingNr (Shipment,Booking)	ShipmentNr (Shipment)	ShipmentRouteNr
25/03/2014 12:00	25/03/2014 13:00	20140321S00062		100	100001065568	100001065568	100001065568
26/03/2014 10:00	26/03/2014 11:00	20140324S00037		100	100001065576	100001065576	100001065576
24/03/2014 15:00	24/03/2014 16:00	AMST048022		100	100001079143	100001079143	100001079143
31/03/2014 07:30	31/03/2014 08:30	1135917		100	100001081155	100001081155	100001081155
31/03/2014 08:30	31/03/2014 09:30	1135918		100	100001081157	100001081157	100001081157
25/03/2014 10:00	25/03/2014 11:00	2 drivers		100	100001081473	100001081473	100001081473
24/03/2014 10:18	24/03/2014 12:33	54296/DBFL		100	100001081875	100001081875	100001081875
24/03/2014 22:00	24/03/2014 23:00	LS Wiesb-Garono		100	100001082337	100001082337	100001082337
26/03/2014 21:00	26/03/2014 22:00	LS Italy		100	100001082342	100001082343	100001082343
26/03/2014 21:00	26/03/2014 22:00	LS Italy		100	100001082342	100001082344	100001082344
26/03/2014 21:00	26/03/2014 22:00	LS Italy		100	100001082342	100001082345	100001082345
29/03/2014 09:00	29/03/2014 10:00	LS Italy		100	100001082346	100001082347	100001082347
29/03/2014 09:00	29/03/2014 10:00	LS Italy		100	100001082346	100001082348	100001082348
29/03/2014 09:00	29/03/2014 10:00	LS Italy		100	100001082346	100001082349	100001082349
25/03/2014 10:00	25/03/2014 11:00	LX6406		100	100001082404	100001082405	100001082405
26/03/2014 03:00	26/03/2014 04:00	LX6407 CHECK		100	100001082404	100001082406	100001082406
29/03/2014 18:00	29/03/2014 19:00	Driscol's/Sotr		100	100001082422	100001082422	100001082422
29/03/2014 18:00	29/03/2014 19:00	Driscol's/Sotr		100	100001082423	100001082423	100001082423
25/03/2014 17:00	25/03/2014 18:00	Gateway FRA		100	100001082424	100001082424	100001082424
28/03/2014 17:00	28/03/2014 18:00	Gateway FRA		100	100001082425	100001082425	100001082425
24/03/2014 21:46	24/03/2014 22:46	240314 KL8434		100	100001082615	100001082615	100001082615
25/03/2014 07:00	25/03/2014 08:00	84-1403-02353 1		100	100001082710	100001082710	100001082710
25/03/2014 08:00	25/03/2014 09:00	84-1403-02353 2		100	100001082716	100001082716	100001082716
22/03/2014 13:16	22/03/2014 15:30	210314 CK1113/A		100	100001082731	100001082732	100001082732
24/03/2014 10:59	24/03/2014 12:44	1135914		100	100001082869	100001082869	100001082869
24/03/2014 22:29	24/03/2014 23:29	240314 KL8818		100	100001082883	100001082883	100001082883
24/03/2014 07:34	24/03/2014 08:00	240314 KL8424		100	100001082884	100001082884	100001082884
25/03/2014 08:15	25/03/2014 09:15	LP 101247		100	100001082942	100001082943	100001082943
26/03/2014 04:06	26/03/2014 05:06	LP 101247		100	100001082942	100001082945	100001082945
24/03/2014 09:16	24/03/2014 12:29	33793		100	100001082954	100001082954	100001082954
25/03/2014 10:00	25/03/2014 11:00	33817		100	100001082956	100001082956	100001082956
25/03/2014 11:00	25/03/2014 12:00	33819		100	100001082957	100001082957	100001082957
24/03/2014 13:05	24/03/2014 14:05	D0000103322		100	100001083071	100001083071	100001083071
24/03/2014 14:59	24/03/2014 15:59	D0000103321		100	100001083072	100001083072	100001083072
24/03/2014 08:00	24/03/2014 11:45	1133477		100	100001083093	100001083093	100001083093
24/03/2014 09:52	24/03/2014 10:02	1135867		100	100001083144	100001083147	100001083147
24/03/2014 10:23	24/03/2014 11:30	1135867		100	100001083144	100001083148	100001083148
24/03/2014 19:31	24/03/2014 20:31	240314 KL8552		100	100001083149	100001083149	100001083149

https://api.flightstats.com/flex/weather/rest/v1/json/all/LAX?codeType=fs
</weatherUrl>
</airport>
<airport>
<fs>JFK</fs>
<iata>JFK</iata>
<icao>KJFK</icao>
<faa>JFK</faa>
<name>John F. Kennedy International Airport</name>
<street1>JFK Airport</street1>
<city>New York</city>
<cityCode>NYC</cityCode>
<stateCode>NY</stateCode>
<postalCode>11430</postalCode>
<countryCode>US</countryCode>
<countryName>United States</countryName>
<regionName>North America</regionName>
<timeZoneRegionName>America/New_York</timeZoneRegionName>
<weatherZone>NYZ178</weatherZone>
<localTime>2013-08-09T04:00:32.489</localTime>
<utcOffsetHours>-4.0</utcOffsetHours>
<latitude>40.642335</latitude>
<longitude>-73.78817</longitude>
<elevationFeet>13</elevationFeet>
<classification>1</classification>
<active>true</active>
<delayIndexUrl>
https://api.flightstats.com/flex/delayindex/rest/v1/json/airports/JFK?codeType=fs
</delayIndexUrl>
<weatherUrl>
https://api.flightstats.com/flex/weather/rest/v1/json/all/JFK?codeType=fs
</weatherUrl>
</airport>
</airports>
</appendix>
<flightTracks>
<flightTrack>
<flightId>305231585</flightId>
<carrierFsCode>AA</carrierFsCode>
<flightNumber>10</flightNumber>
<callsign>AAL10</callsign>
<departureAirportFsCode>LAX</departureAirportFsCode>
<arrivalAirportFsCode>JFK</arrivalAirportFsCode>
<departureDate>
<dateLocal>2013-08-07T21:30:00.000</dateLocal>
<dateUtc>2013-08-08T04:30:00.000Z</dateUtc>
</departureDate>
<equipment>B762</equipment>
<bearing>229.53397527577076</bearing>
<heading>228.69263064295163</heading>
<positions>
<position>
<lon>-73.7300033569336</lon>
<lat>40.68000030517578</lat>
<speedMph>137</speedMph>
<altitudeFt>800</altitudeFt>
<source>ASDI</source>
<date>2013-08-08T09:14:56.000Z</date>
</position>
<position>
<lon>-73.69999694824219</lon>
<lat>40.70000076293945</lat>
<speedMph>143</speedMph>
<altitudeFt>1400</altitudeFt>
<source>ASDI</source>
<date>2013-08-08T09:13:56.000Z</date>
</position>
<position>
<lon>-73.69999694824219</lon>
<lat>40.70000076293945</lat>
<speedMph>157</speedMph>
<altitudeFt>1500</altitudeFt>
<source>ASDI</source>
<date>2013-08-08T09:13:56.000Z</date>
</position>
<position>
<lon>-73.66999816894531</lon>
<lat>40.72999954223633</lat>
<speedMph>163</speedMph>
<altitudeFt>2000</altitudeFt>
<source>ASDI</source>
<date>2013-08-08T09:12:55.000Z</date>
</position>

IEEE Standard 1849-2016 for eXtensible Event Stream (XES)




```
- <log xes.version="1.0" xes.features="nested-attributes" openxes.version="1.0RC7">
    <extension name="Lifecycle" prefix="lifecycle" uri="http://www.xes-standard.org/lifecycle.xesext"/>
    <extension name="Time" prefix="time" uri="http://www.xes-standard.org/time.xesext"/>
    <extension name="Concept" prefix="concept" uri="http://www.xes-standard.org/concept.xesext"/>
    <classifier name="Event Name" keys="concept:name"/>
    <string key="concept:name" value="Email Log"/>
    <string key="lifecycle:model" value="standard"/>
  - <trace>
      <string key="concept:name" value="dc.claudio@gmail.com/SM4All"/>
    - <event>
        <string key="lifecycle:transition" value="complete"/>
        <string key="concept:name" value="send agenda"/>
        <date key="time:timestamp" value="2009-07-09T17:44:59Z"/>
      </event>
    - <event>
        <string key="lifecycle:transition" value="complete"/>
        <string key="concept:name" value="send meeting"/>
        <date key="time:timestamp" value="2009-07-14T22:24:43Z"/>
      </event>
    - <event>
        <string key="lifecycle:transition" value="complete"/>
        <string key="concept:name" value="send draft"/>
        <date key="time:timestamp" value="2009-09-11T17:05:50Z"/>
      </event>
    - <event>
        <string key="lifecycle:transition" value="complete"/>
        <string key="concept:name" value="send draft"/>
        <date key="time:timestamp" value="2009-09-14T10:21:42Z"/>
      </event>
    - <event>
        <string key="lifecycle:transition" value="complete"/>
        <string key="concept:name" value="send draft"/>
        <date key="time:timestamp" value="2009-10-12T21:31:49Z"/>
      </event>
    - <event>
        <string key="lifecycle:transition" value="complete"/>
        <string key="concept:name" value="write deliverable"/>
        <date key="time:timestamp" value="2010-01-12T23:16:34Z"/>
      </event>
    - <event>
        <string key="lifecycle:transition" value="complete"/>
        <string key="concept:name" value="send report"/>
        <date key="time:timestamp" value="2010-01-13T16:00:58Z"/>
      </event>
    - <event>
        <string key="lifecycle:transition" value="complete"/>
        <string key="concept:name" value="write deliverable"/>
```

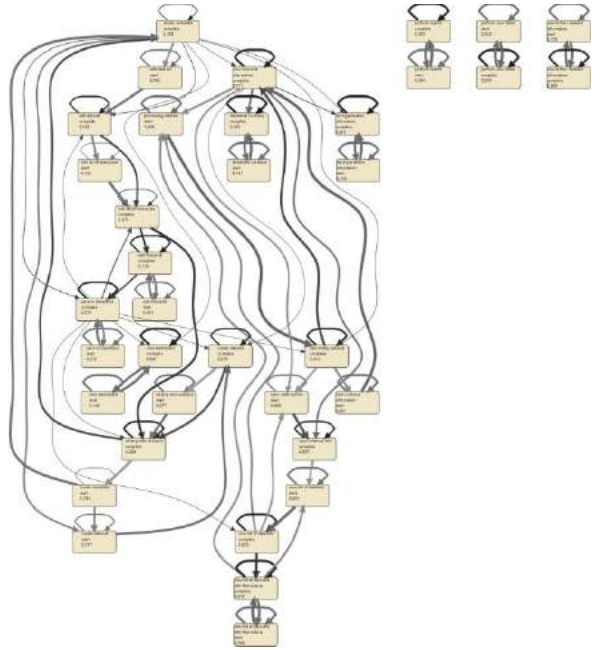

http://www.xes-standard.org/

Process analytics: Measure the process performance

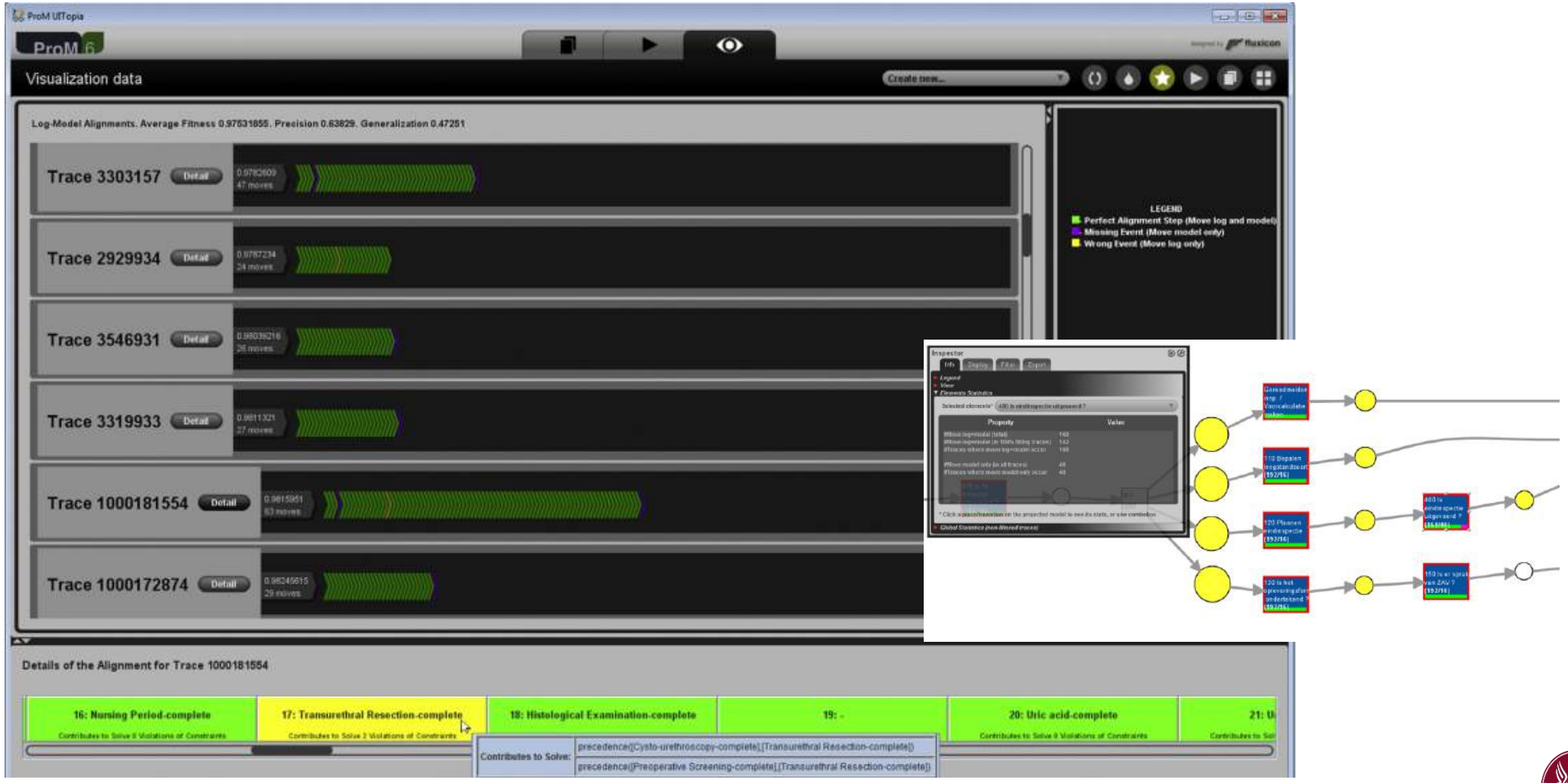
Slide courtesy of Matej Puchovsky

Automated process discovery: Understanding the process behind the data

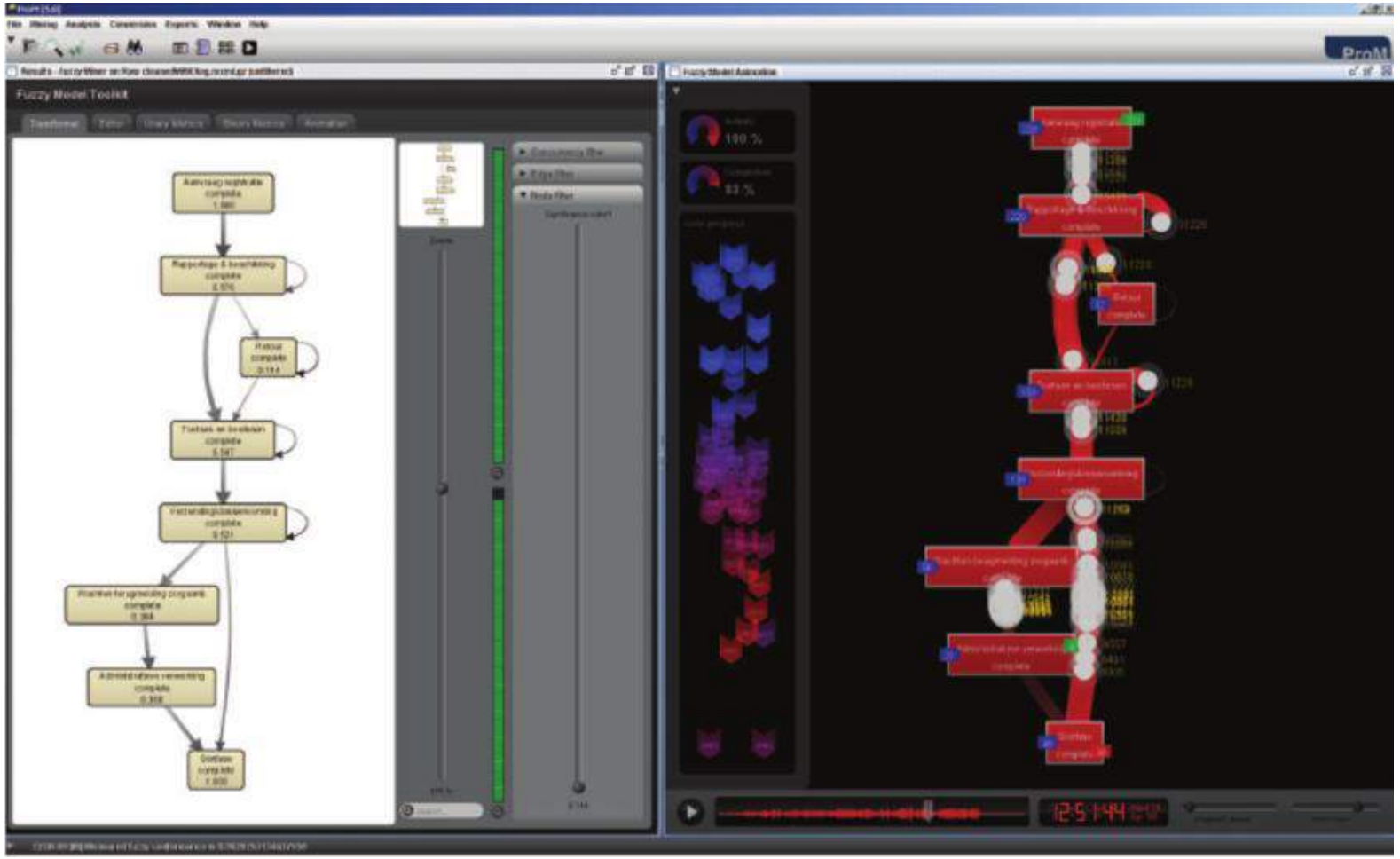
```
<log xes.version="1.0" xes.features="nested-attributes" openxes.version="1.0RC7">
    <string key="concept:name" value="Klassifikation_v4.csv"/>
    <trace>
        <string key="visitIP" value="178.114.0.0"/>
        <string key="concept:name" value="77095"/>
        <string key="visitorId" value="e8a0cfadda07194d"/>
        <event>
            <string key="concept:instance" value="1"/>
            <string key="lifecycle:transition" value="start"/>
            <string key="concept:name" value="outlink"/>
            <date key="time:timestamp" value="2014-07-08T11:09:43.000+02:00"/>
        </event>
        <event>
            <string key="siteSearchKeyword"/>
            <string key="country" value="□sterreich"/>
            <string key="pageTitle"/>
            <string key="visitorType" value="returning"/>
            <string key="visitDurationPretty" value="2 Minuten 45s"/>
            <string key="customVariables"/>
            <string key="concept:name" value="outlink"/>
            <string key="visitServerHour" value="9"/>
            <string key="providerName" value="unbekannt"/>
            <string key="longitude" value="13.333000"/>
            <string key="searches" value="0"/>
            <string key="referrerTypeName" value="Direkte Zugriffe"/>
            <string key="visitorTypeIcon" value="plugins/Live/images/returningVisitor.gif"/>
            <string key="pageId" value="184467"/>
            <string key="serverTimePrettyFirstAction" value="11:12:17"/>
            <string key="providerUrl"/>
            <date key="time:timestamp" value="2014-07-08T11:12:22.000+02:00"/>
            <string key="serverDate" value="08.07.2014"/>
            <string key="generationTime"/>
            <string key="lastActionTimestamp" value="1404810901"/>
            <string key="visitEcommerceStatusIcon"/>
            <string key="referrerSearchEngineUrl"/>
            <string key="visitLocalTime" value="11:12:04"/>
            <string key="region"/>
            <string key="actions" value="3"/>
            <string key="lastActionDateTime" value="08.07.2014 09:15"/>
            <string key="continent" value="Europa"/>
            <string key="visitDuration" value="165"/>
            <int key="daysSinceLastVisit" value="1"/>
            <string key="pageIdAction" value="13130"/>
            <string key="city"/>
            <string key="latitude" value="47.333000"/>
            <string key="icon" value="plugins/Morpheus/images/link.gif"/>
            <string key="serverDatePrettyFirstAction" value="Dienstag, 8. Juli 2014"/>
            <string key="referrerKeyword"/>
            <int key="daysSinceFirstVisit" value="1"/>
            <string key="serverTimePretty" value="08.7.2014 11:12:22/11:15:01"/>
            <string key="regionCode"/>
            <string key="visitCount" value="2"/>
            <string key="referrerType" value="direct"/>
            <string key="countryFlag" value="plugins/UserCountry/images/flags/at.png"/>
```



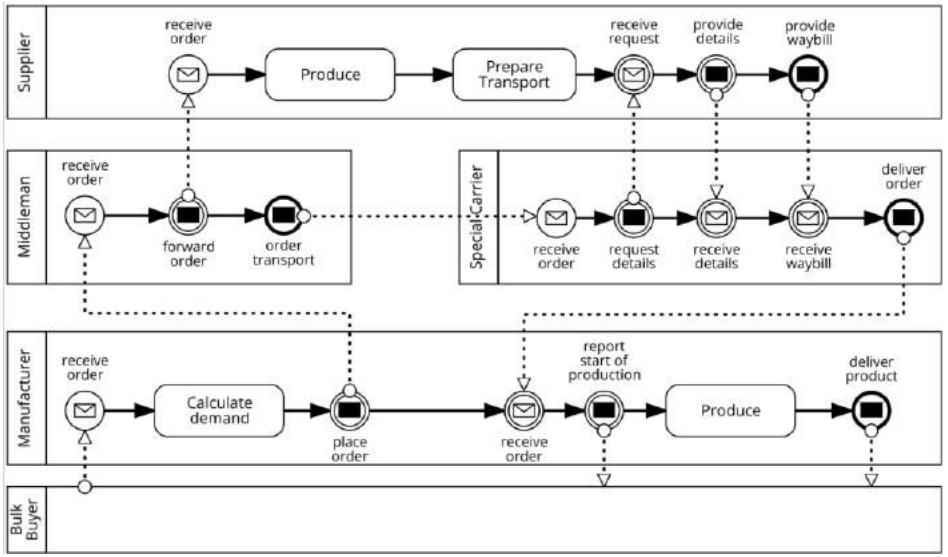
Conformance checking: Detecting deviations and bottlenecks

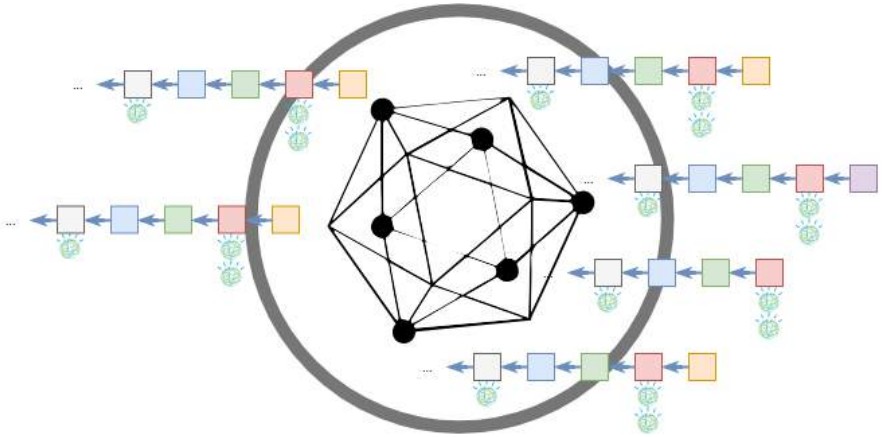


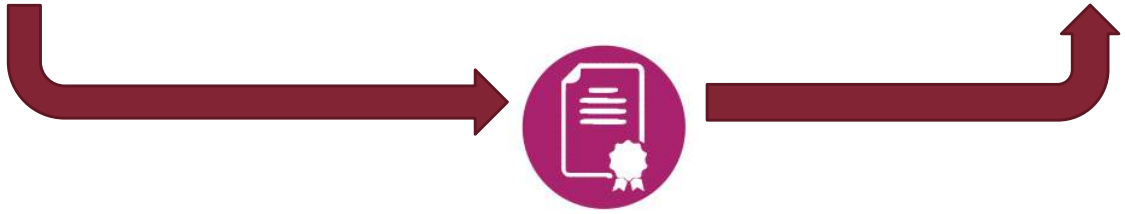
Process enhancement:
Improve the process based on the data



Executing inter-organisational processes on the Blockchain: A model-driven approach

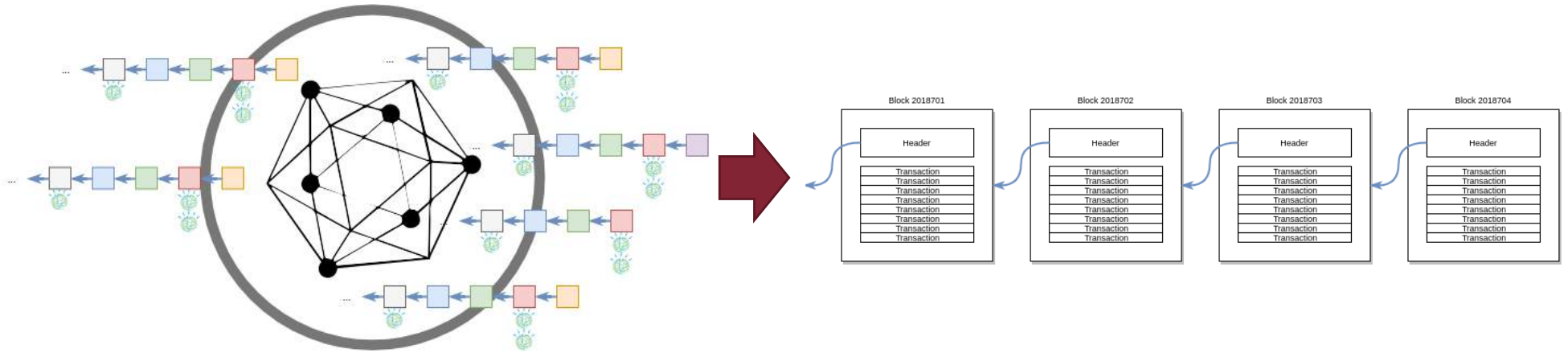




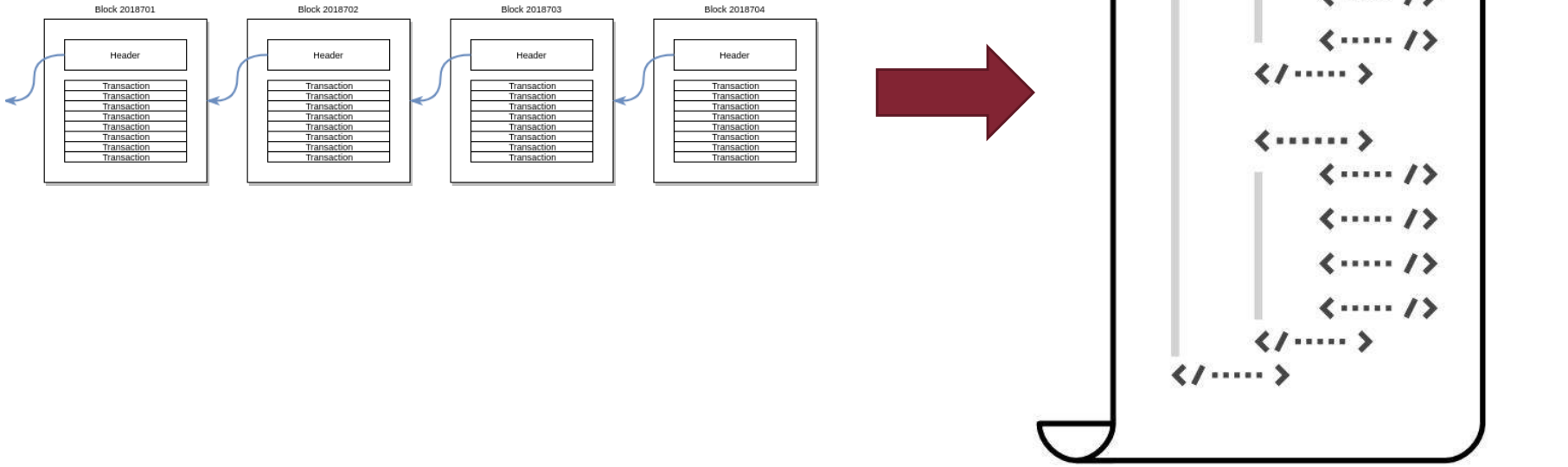


D.C, Cecconi, Dumas et al. Blockchain support for collaborative business processes. Informatik Spektrum pp. 1–9 (May 2019)

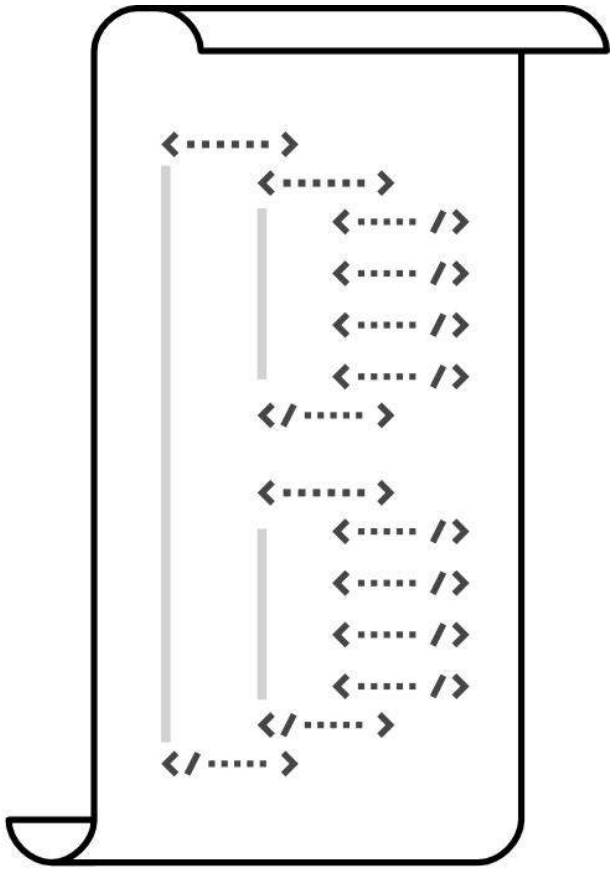
From ledgers to event logs

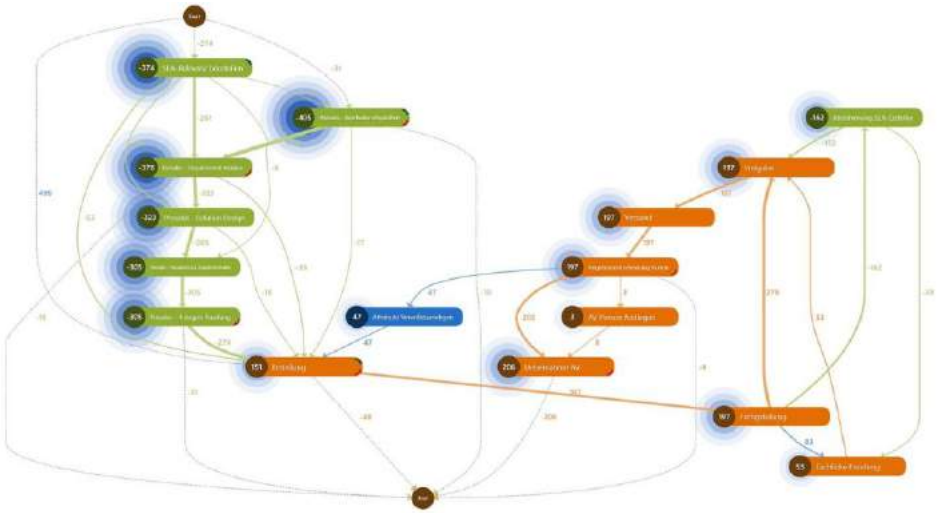


From ledgers to event logs



From event logs to process analytics





The journey

Mining the Blockchain for processes

In 2012…

DOI:10.1145/2076450.2076466

Article development led by acmqueue
queue.acm.org

Logs contain a wealth of information to help manage systems.

BY ADAM OLINER, ARCHANA GANAPATHI, AND WEI XU

Advances and Challenges in Log Analysis

COMPUTER-SYSTEM LOGS provide a glimpse into the states of a running system. Instrumentation occasionally generates short messages that are collected in a system-specific log. The content and format of logs can vary widely from one system to another and even among components within a system. A printer driver

1979…

Many logs are intended to facilitate debugging. As Brian Kernighan wrote in *Unix for Beginners* in 1979, "The most effective debugging tool is still careful thought, coupled with judiciously placed print statements." Although today's programs are orders of magnitude more complex than those of 30 years ago, many people still use `printf` to log to console or local disk, and use some combination of manual inspection and regular expressions to locate specific messages or patterns.

And we are not yet over it.

Oliner et al. Advances and Challenges in Log Analysis. Communications of the ACM, 2012

Logs bear valuable insights

Mining Specifications

Glenn Ammons
Dept. of Computer Sciences
University of Wisconsin
Madison, Wisconsin, USA
ammons@cs.wisc.edu

Rastislav Bodík
Dept. of Computer Sciences
University of Wisconsin
Madison, Wisconsin, USA
bodik@cs.wisc.edu

James R. Larus
Microsoft Research
One Microsoft Way
Redmond, Washington, USA
larus@microsoft.com

ABSTRACT

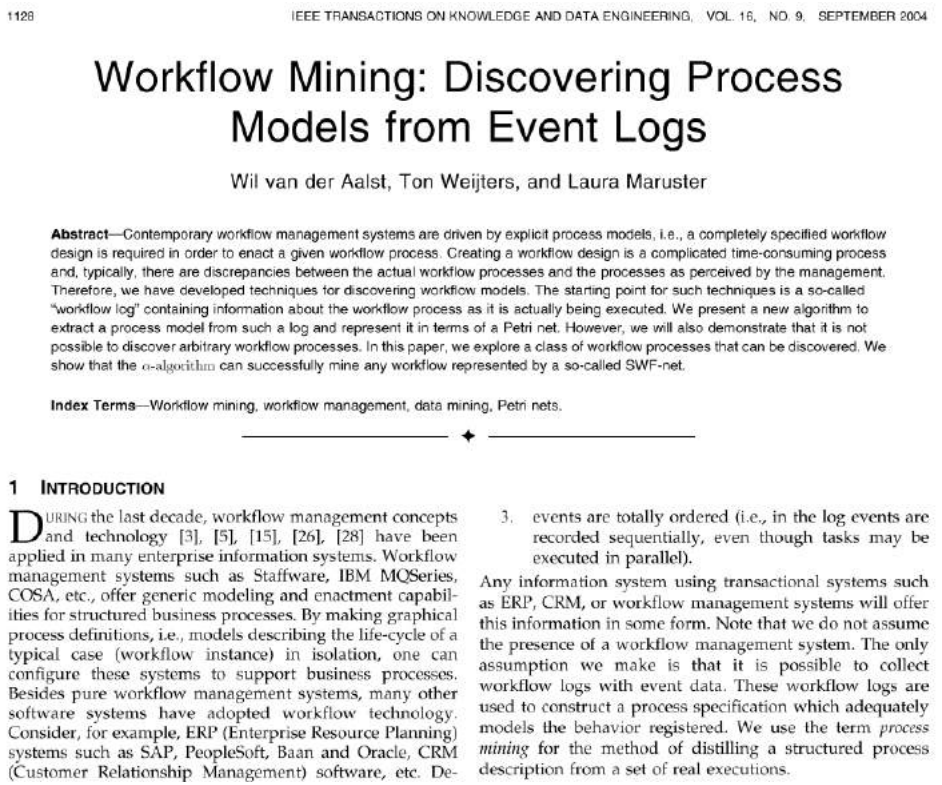
Program verification is a promising approach to improving program quality, because it can search all possible program executions for specific errors. However, the need to formally describe correct behavior or errors is a major barrier to the widespread adoption of program verification, since programmers historically have been reluctant to write formal specifications. Automating the process of formulating specifications would remove a barrier to program verification and enhance its practicality.

This paper describes *specification mining*, a machine learning approach to discovering formal specifications of the protocols that code must obey when interacting with an application program interface or abstract data type. Starting from the assumption that a working program is well enough debugged to reveal strong hints of correct protocols, our tool infers a specification by observing program execution and concisely summarizing the frequent interaction patterns as state machines that capture both temporal and data dependences. These state machines can be examined by a programmer, to refine the specification and identify errors, and can be utilized by automatic verification tools, to find bugs.

These tools, in general, statically compute an approximation of a program's possible dynamic behaviors and compare it against a specification of correct behavior. These specifications often are easy to develop for language-specific properties—such as avoiding null dereferences and staying within array bounds. Even when language properties are more difficult to express and check, they potentially apply to every program written in the language, so an investment in a verification tool can be amortized easily.

On the other hand, specifications particular to a program, say of its abstractions or datatypes, may be difficult and expensive to develop because of the complexity of these mechanisms and the limited number of people who understand them. Also, as these specifications may apply to only one program, their benefits are correspondingly reduced. Program verification is unlikely to be widely used without cheaper and easier ways to formulate specifications.

This paper explores one approach to automating much of the process of producing specifications. This approach, called *specification mining*, discovers some of the temporal and data-dependence relationships that a program follows when it interacts with an application programming interface (API) or abstract datatype (ADT).



Workflow Mining: Discovering Process Models from Event Logs

Wil van der Aalst, Ton Weijters, and Laura Maruster

Abstract—Contemporary workflow management systems are driven by explicit process models, i.e., a completely specified workflow design is required in order to enact a given workflow process. Creating a workflow design is a complicated time-consuming process and, typically, there are discrepancies between the actual workflow processes and the processes as perceived by the management. Therefore, we have developed techniques for discovering workflow models. The starting point for such techniques is a so-called "workflow log" containing information about the workflow process as it is actually being executed. We present a new algorithm to extract a process model from such a log and represent it in terms of a Petri net. However, we will also demonstrate that it is not possible to discover arbitrary workflow processes. In this paper, we explore a class of workflow processes that can be discovered. We show that the α-algorithm can successfully mine any workflow represented by a so-called SWF-net.

Index Terms—Workflow mining, workflow management, data mining, Petri nets.

1 Introduction

During the last decade, workflow management concepts and technology [3], [5], [15], [26], [28] have been applied in many enterprise information systems. Workflow management systems such as Staffware, IBM MQSeries, COSA, etc., offer generic modeling and enactment capabilities for structured business processes. By making graphical process definitions, i.e., models describing the life-cycle of a typical case (workflow instance) in isolation, one can configure these systems to support business processes. Besides pure workflow management systems, many other software systems have adopted workflow technology. Consider, for example, ERP (Enterprise Resource Planning) systems such as SAP, PeopleSoft, Baan and Oracle, CRM (Customer Relationship Management) software, etc. De-

3. events are totally ordered (i.e., in the log events are recorded sequentially, even though tasks may be executed in parallel).

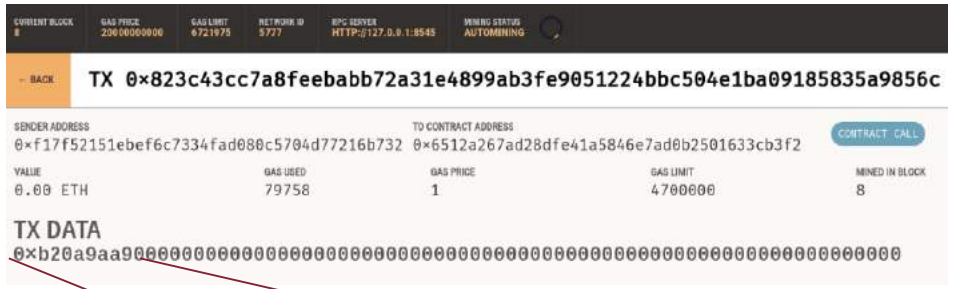
Any information system using transactional systems such as ERP, CRM, or workflow management systems will offer this information in some form. Note that we do not assume the presence of a workflow management system. The only assumption we make is that it is possible to collect workflow logs with event data. These workflow logs are used to construct a process specification which adequately models the behavior registered. We use the term *process mining* for the method of distilling a structured process description from a set of real executions.

Ammons et al. 2002; van der Aalst et al. 2004

Understanding the enacted task

Blockchain-Based Traceability
of Inter-organisational Business Processes

Claudio Di Ciccio, Alessio Cecconi, Jan Mendling, Dominik Felix,
Dominik Haas, Daniel Lilek, Florian Riel, Andreas Rumpl, and Philipp Uhlig



Example

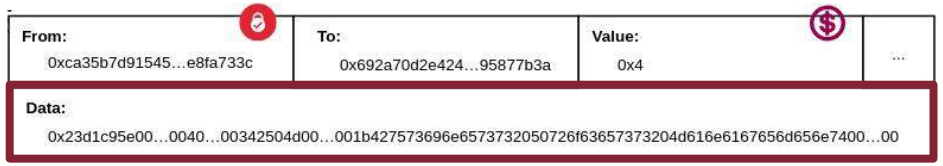
- Function signature:

```
function receive_order(uint256) public
```

- Corresponding Application Binary Interface (ABI):

```
… { "constant":false,
    "inputs":[{"name":"workitemId",
    "type":"uint256"}],
    "name":"receive_order",
    "outputs":[],
    "payable":false,
    "stateMutability":"nonpayable",
                " } …
```

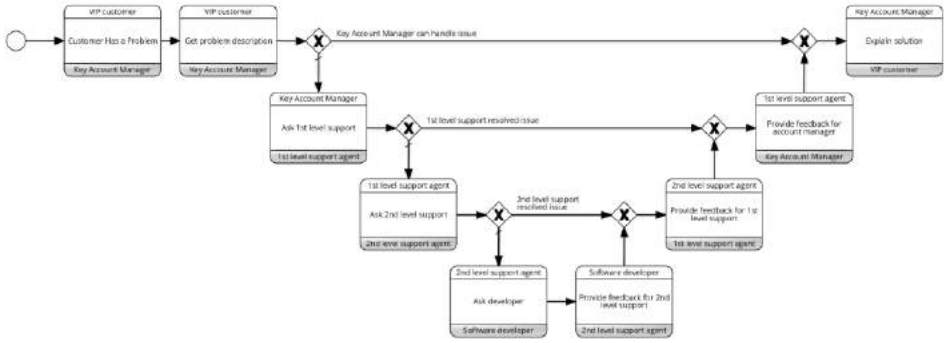
SHA3("**receive_order**")=0x**b20a9aa9**619b9dc0c6a76bef0c51350699afd93b05757350f8c20de71bc75e18



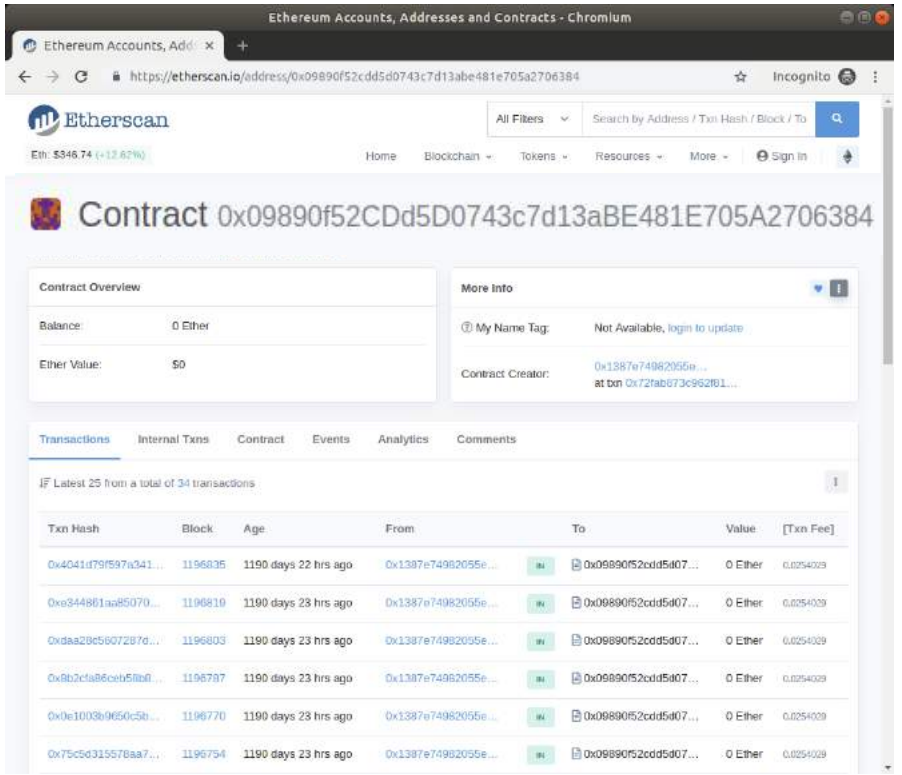
D.C, Cecconi, Mendling et al: Blockchain-based Traceability of Inter-organisational Business Processes. In: Proc. of BMSD 2018.

Case study

Original process

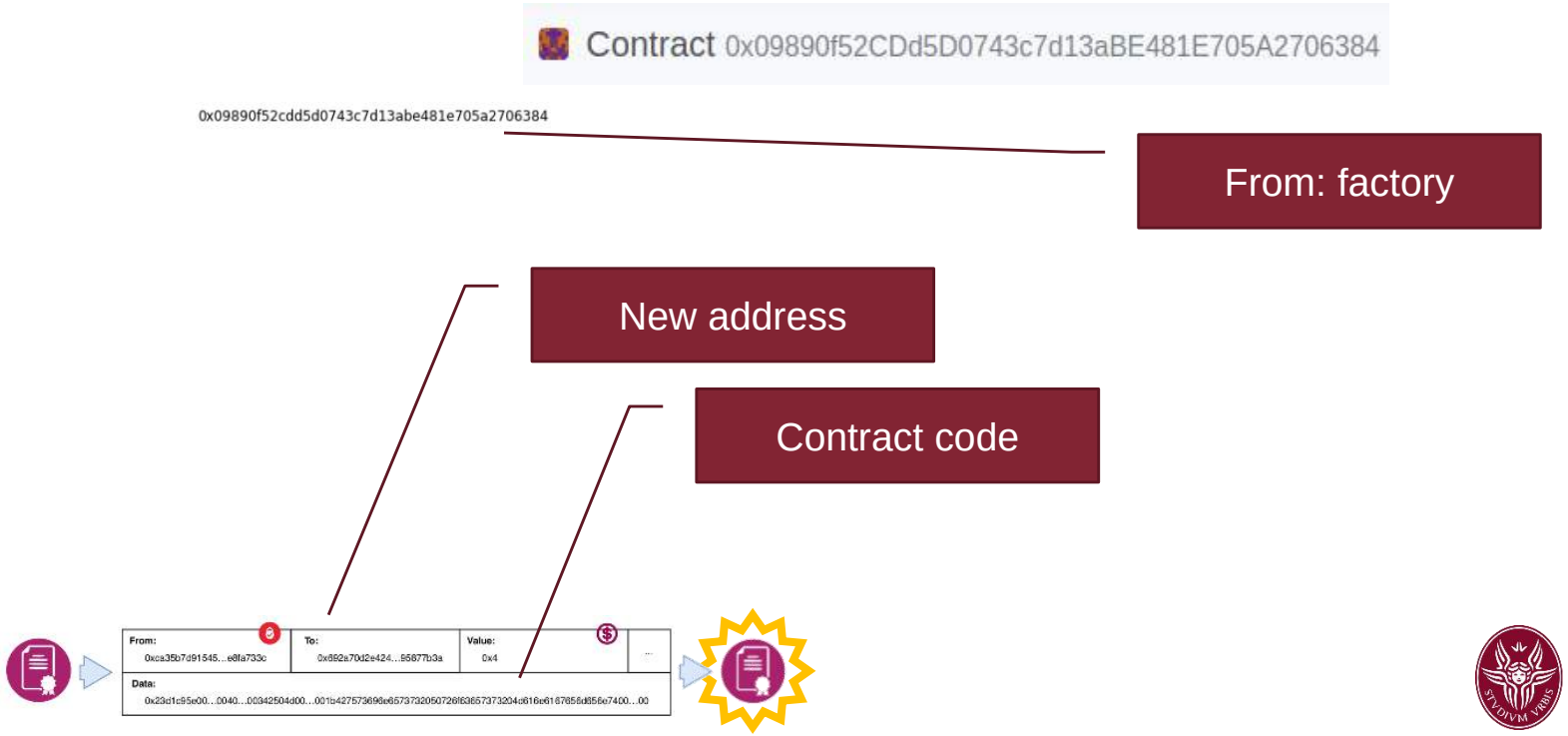


Factory contract

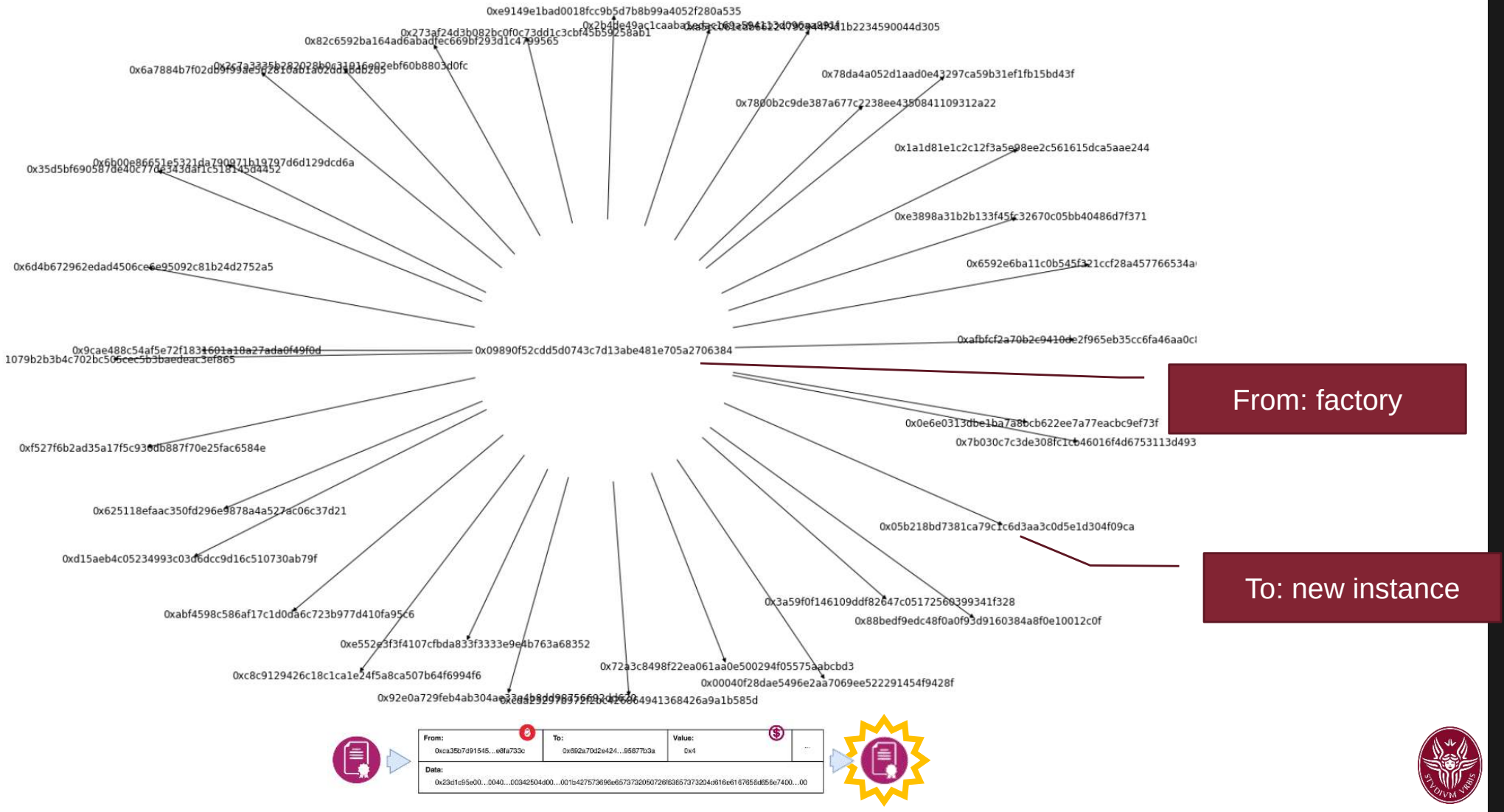


Weber et al.: Untrusted business process monitoring and execution using blockchain. In: Proc. of BPM, 329-347. Springer, 2016.

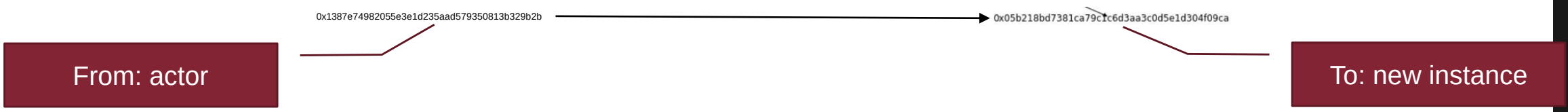
Intercepting new instantiations



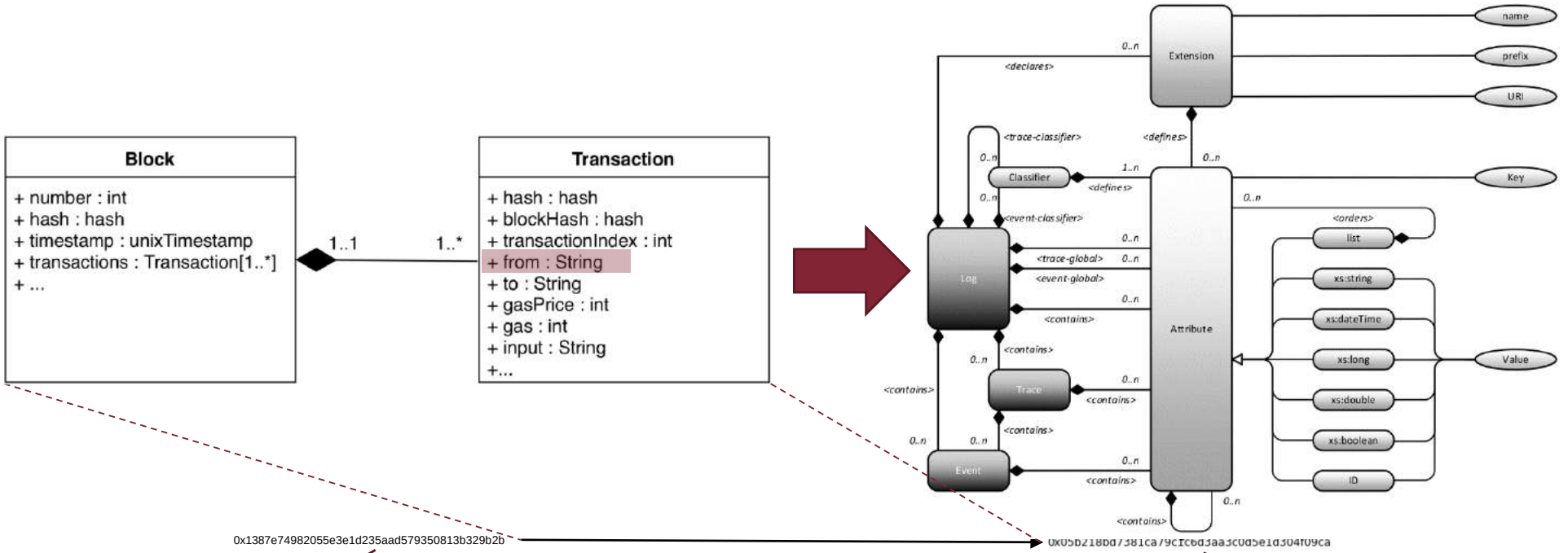
Intercepting new instantiations



Intercepting calls to process contract



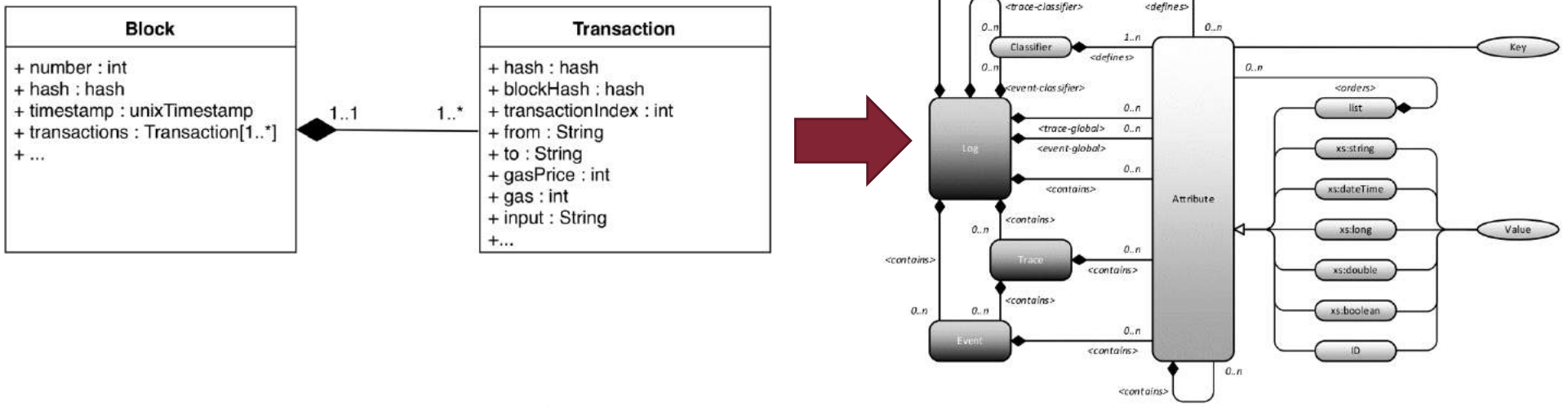
Intercepting calls to process contract



From: actor

To: new instance

From transactions to events



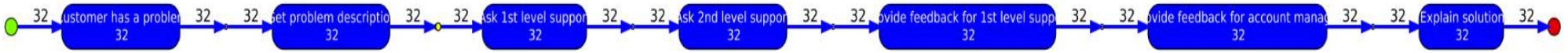
Event log attribute	Blockchain data field
Case ID	Transaction.to
Activity ID	Transaction.input[2:10] (function selector)
Event ID	Transaction.hash
Activity label	Reverse-engineered from contract ABI and function selector
Event timestamp	Block.timestamp (plus sorting by Transaction.index)
Event cost	Transaction.gas
Event resource	Transaction.from

https://gitlab.com/MacOS/extracting-event-logs-from-process-data-on-the-Blockchain/

Event log

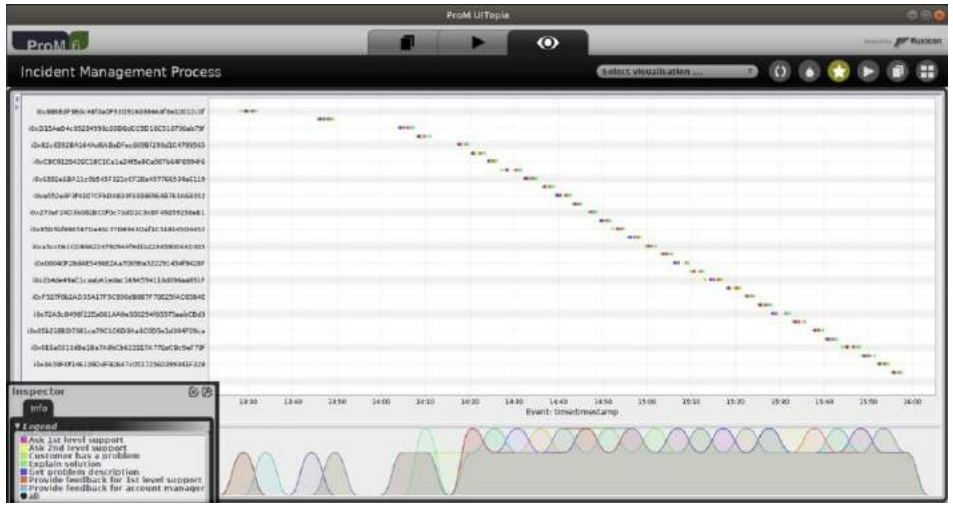
```
<log openxes.version="1.0RC7" xes.features="" xes.version="1.0">
        <extension name="Concept" prefix="concept" uri="http://www.xes-standard.org/concept.xesext"/>
        <extension name="Identity" prefix="identity" uri="http://www.xes-standard.org/identity.xesext"/>
        <extension name="Organizational" prefix="org" uri="http://www.xes-standard.org/org.xesext"/>
        <extension name="Lifecycle" prefix="lifecycle" uri="http://www.xes-standard.org/lifecycle.xesext"/>
        <extension name="Time" prefix="time" uri="http://www.xes-standard.org/time.xesext"/>
        <extension name="Cost" prefix="cost" uri="http://www.xes-standard.org/cost.xesext"/>
        <string key="concept:name" value="Incident Management Process"/>
        <string key="lifecycle:model" value="standard"/>
        <trace>
                <string key="concept:name" value="i0x00040F28dAE5496E2Aa7069Ee522291454f9428F"/>
                <string key="id" value="i0x00040F28dAE5496E2Aa7069Ee522291454f9428F"/>
                <event>
                        <string key="id" value="i0x29e663ef7b84c54f0edc3e3fd143499cbabb574f262cf99c669b4382156d7d53"/>
                        <string key="activity_id" value="efe73dcb"/>
                        <string key="concept:name" value="Customer has a problem"/>
                        <date key="time:timestamp" value="2016-03-22T14:06:22.000Z"/>
                        <string key="org:resource" value="Key Account Manager"/>
                        <string key="lifecycle:transition" value="complete"/>
                        <int key="blockNo" value="1196605"/>
                        <string key="from" value="0x1387E74982055e3e1d235AaD579350813B329b2B"/>
                        <string key="transactionData" value="0xefe73dcb"/>
                </event>
                <event>
                        <string key="id" value="i0x4642478f99a5d70d2b7022384b5770ca22c029f72c09e00704295c48cb27937b"/>
                        <string key="activity_id" value="b46d63fa"/>
                        <string key="concept:name" value="Get problem description"/>
                        <date key="time:timestamp" value="2016-03-22T14:06:35.000Z"/>
                        <string key="org:resource" value="Key Account Manager"/>
                        <string key="lifecycle:transition" value="complete"/>
                        <int key="blockNo" value="1196607"/>
                        <string key="from" value="0x1387E74982055e3e1d235AaD579350813B329b2B"/>
                        <string key="transactionData" value="0xb46d63fa0000000000000000000000000000000000000000000000000000000000000002"/>
                </event>
                <event>
                        <string key="id" value="i0x449cb40c931b8db89d28be73e7498a7fd9c4463336457c85c1f4f5de72cb8a8f"/>
                        <string key="activity_id" value="dfe3c3ec"/>
                        <string key="concept:name" value="Ask 1st level support"/>
                        <date key="time:timestamp" value="2016-03-22T14:07:20.000Z"/>
                        <string key="org:resource" value="Key Account Manager"/>
                        <string key="lifecycle:transition" value="complete"/>
                        <int key="blockNo" value="1196609"/>
                        <string key="from" value="0x1387E74982055e3e1d235AaD579350813B329b2B"/>
```

https://gitlab.com/MacOS/extracting-event-logs-from-process-data-on-the-blockchain/blob/master/incident_management_process.xes

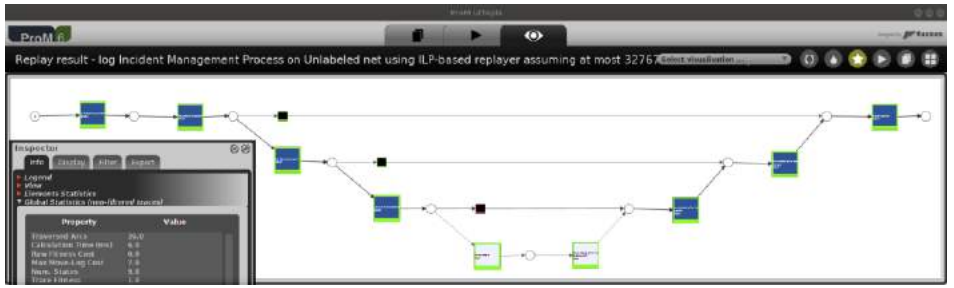


Process analytics

Log inspection



Conformance checking
Hand-over analysis



In 2019…

Mühlberger et al. 2019

Extracting Event Logs for Process Mining from Data Stored on the Blockchain

Roman Mühlberger[1], Stefan Bachhofner[1], Claudio Di Ciccio[1(✉)], Luciano García-Bañuelos[2,3], and Orlenys López-Pintado[2]

[1] Vienna University of Economics and Business, Vienna, Austria
roman@muehlberger.eu.com, {stefan.bachhofner,claudio.di.ciccio}@wu.ac.at
[2] University of Tartu, Tartu, Estonia
orlenyslp@ut.ee
[3] Tecnológico de Monterrey, Monterrey, Mexico
luciano.garcia@tec.mx

Abstract. The integration of business process management with blockchains across organisational borders provides a means to establish transparency of execution and auditing capabilities. To enable process analytics, though, non-trivial extraction and transformation tasks are necessary on the raw data stored in the ledger. In this paper, we describe our approach to retrieve process data from an Ethereum blockchain ledger and subsequently convert those data into an event log formatted according to the IEEE Extensible Event Stream (XES) standard. We show a proof-of-concept software artefact and its application on a data set produced by the smart contracts of a process execution engine stored on the public Ethereum blockchain network.

Keywords: Ethereum · Process discovery · Process monitoring · Process conformance

Klinkmüller et al. 2019

Mining Blockchain Processes: Extracting Process Mining Data from Blockchain Applications

Christopher Klinkmüller[1(✉)], Alexander Ponomarev[1], An Binh Tran[1], Ingo Weber[2,3], and Wil van der Aalst[4]

[1] Data61, CSIRO, Level 5, 13 Garden Street, Eveleigh, NSW 2015, Australia
{christopher.klinkmuller,alex.ponomarev,anbinh.tran}@data61.csiro.au
[2] Technische Universitaet Berlin, Berlin, Germany
ingo.weber@tu-berlin.de
[3] University of New South Wales, Sydney, NSW 2052, Australia
[4] RWTH Aachen University, Aachen, Germany
wvdaalst@pads.rwth-aachen.de

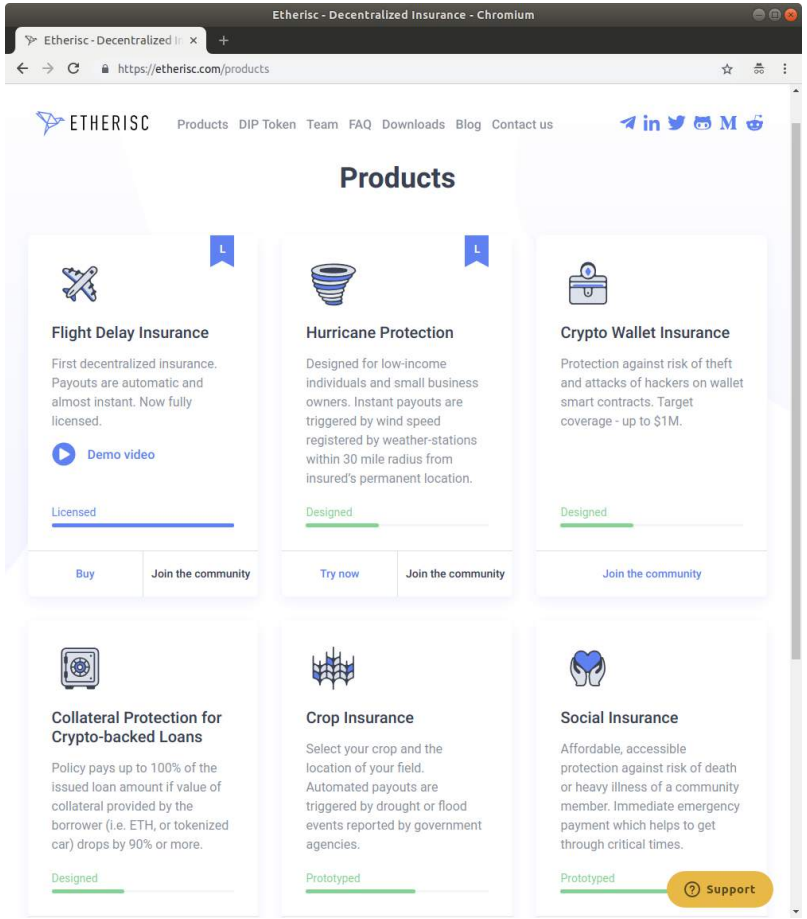
Abstract. Blockchain technology has been gaining popularity as a platform for developing decentralized applications and executing cross-organisational processes. However, extracting data that allows analysing the process view from blockchains is surprisingly hard. Therefore, blockchain data are rarely used for process mining. In this paper, we propose a framework for alleviating that pain. The framework comprises three main parts: a manifest specifying how data is logged, an extractor for retrieving data (structured according to the XES standard), and a generator that produces logging code to support smart contract developers. Among others, we propose a convenient way to encode logging data in a compact form, to achieve relatively low cost and high throughput for on-chain logging. The proposal is evaluated with logs created from generated logging code, as well as with existing blockchain applications that do not make use of the proposed code generator.

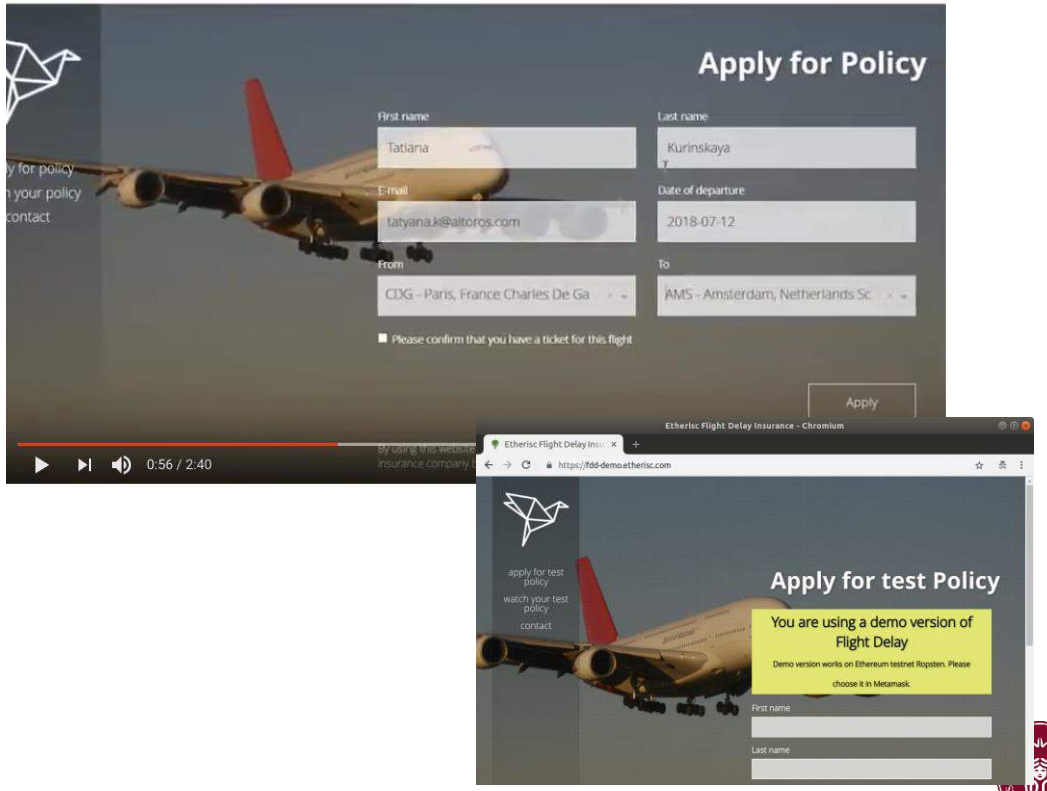
Keywords: Process mining · Blockchain · Smart contracts · Logging · XES

How about the real world?

Oracles: From on-chain to off-chain and vice versa

Etherisc


Etherisc - Decentralized Insurance - Chromium
https://etherisc.com/products
ETHERISC
Products DIP Token Team FAQ Downloads Blog Contact us
Products
Flight Delay Insurance
First decentralized insurance. Payouts are automatic and almost instant. Now fully licensed.
Demo video
Licensed
Buy
Join the community
Hurricane Protection
Designed for low-income individuals and small business owners. Instant payouts are triggered by wind speed registered by weather-stations within 30 mile radius from insured's permanent location.
Designed
Try now
Join the community
Crypto Wallet Insurance
Protection against risk of theft and attacks of hackers on wallet smart contracts. Target coverage - up to $1M.
Designed
Join the community
Collateral Protection for Crypto-backed Loans
Policy pays up to 100% of the issued loan amount if value of collateral provided by the borrower (i.e. ETH, or tokenized car) drops by 90% or more.
Designed
Crop Insurance
Select your crop and the location of your field. Automated payouts are triggered by drought or flood events reported by government agencies.
Prototyped
Social Insurance
Affordable, accessible protection against risk of death or heavy illness of a community member. Immediate emergency payment which helps to get through critical times.
Prototyped
Support


Apply for Policy
First name
Tatiana
Last name
Kurinskaya
E-mail
tatyana.k@altoros.com
Date of departure
2018-07-12
From
CDG - Paris, France Charles De Ga
To
AMS - Amsterdam, Netherlands Sc
Please confirm that you have a ticket for this flight
Apply
0:56 / 2:40
Etherisc Flight Delay Insurance - Chromium
https://fdd-demo.etherisc.com
Apply for test Policy
You are using a demo version of Flight Delay
Demo version works on Ethereum testnet Ropsten. Please choose it in Metamask.
First name
Last name

Flight delay insurance: the FlightDelayPayout contract

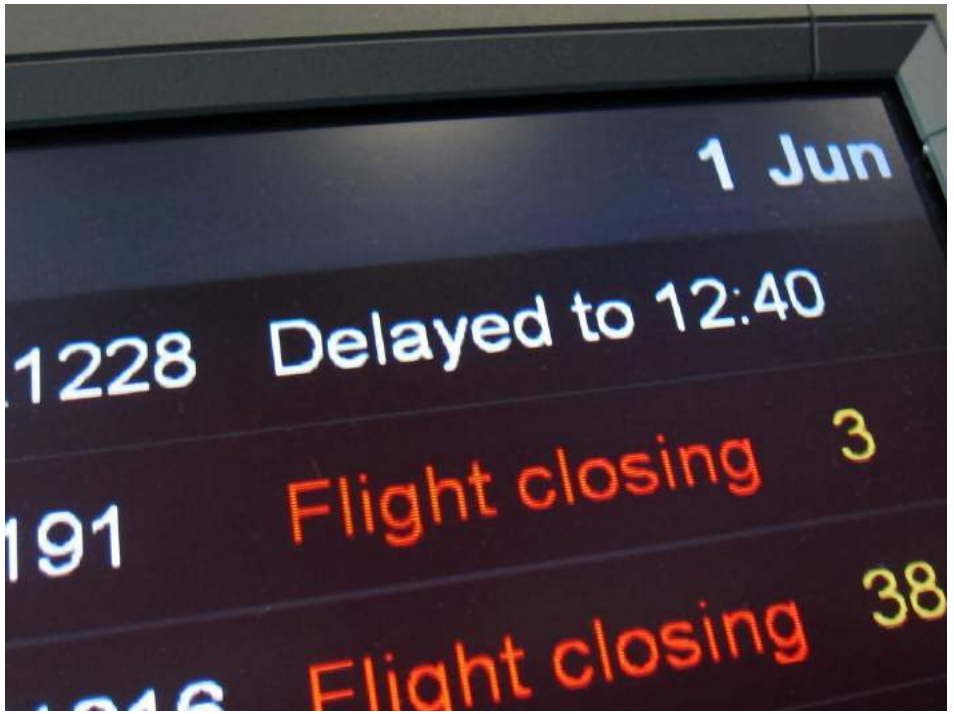
flightDelay/FlightDelayPayout.sol at master · etherisc/flightDelay · GitHub - Chromium

GitHub, Inc. [US] | https://github.com/etherisc/flightDelay/blob/master/contracts/FlightDelayPayout.sol

```
100     /*
101      * @dev Oraclize callback. In an emergency case, we can call this directly from FD.Emergency Account.
102      * @param _queryId
103      * @param _result
104      * @param _proof
105      */
106     function __callback(bytes32 _queryId, string _result, bytes _proof) public onlyOraclizeOf(getContract('FD.Emergency')) {
107
108         var (policyId, oraclizeTime) = FD_DB.getOraclizeCallback(_queryId);
109         LogOraclizeCallback(policyId, _queryId, _result, _proof);
110
111         // check if policy was declined after this callback was scheduled
112         var state = FD_DB.getPolicyState(policyId);
113         require(uint8(state) != 5);
114
115         bytes32 riskId = FD_DB.getRiskId(policyId);
116
117 // --> debug-mode
118 //          LogBytes32("riskId", riskId);
119 // <-- debug-mode
120
121         var slResult = _result.toSlice();
122
123         if (bytes(_result).length == 0) { // empty Result
124             if (FD_DB.checkTime(_queryId, riskId, 180 minutes)) {
125                 LogPolicyManualPayout(policyId, "No Callback at +120 min");
126                 return;
127             } else {
128                 schedulePayoutOraclizeCall(policyId, riskId, oraclizeTime + 45 minutes);
129             }
130         } else {
131             // first check status
132             // extract the status field:
133             slResult.find("\"".toSlice()).beyond("\"".toSlice());
134             slResult.until(slResult.copy().find("\"".toSlice()));
135             bytes1 status = bytes(slResult.toString())[0];  // s = L
136             if (status == "C") {
137                 // flight cancelled --> payout
138                 payOut(policyId, 4, 0);
139                 return;
140             } else if (status == "D") {
141                 // flight diverted --> payout
142                 payOut(policyId, 5, 0);
143                 return;
144             } else if (status != "L" && status != "A" && status != "C" && status != "D") {
145                 LogPolicyManualPayout(policyId, "Unprocessable status");
146                 return;
147             }
148
149             // process the rest of the response:
```

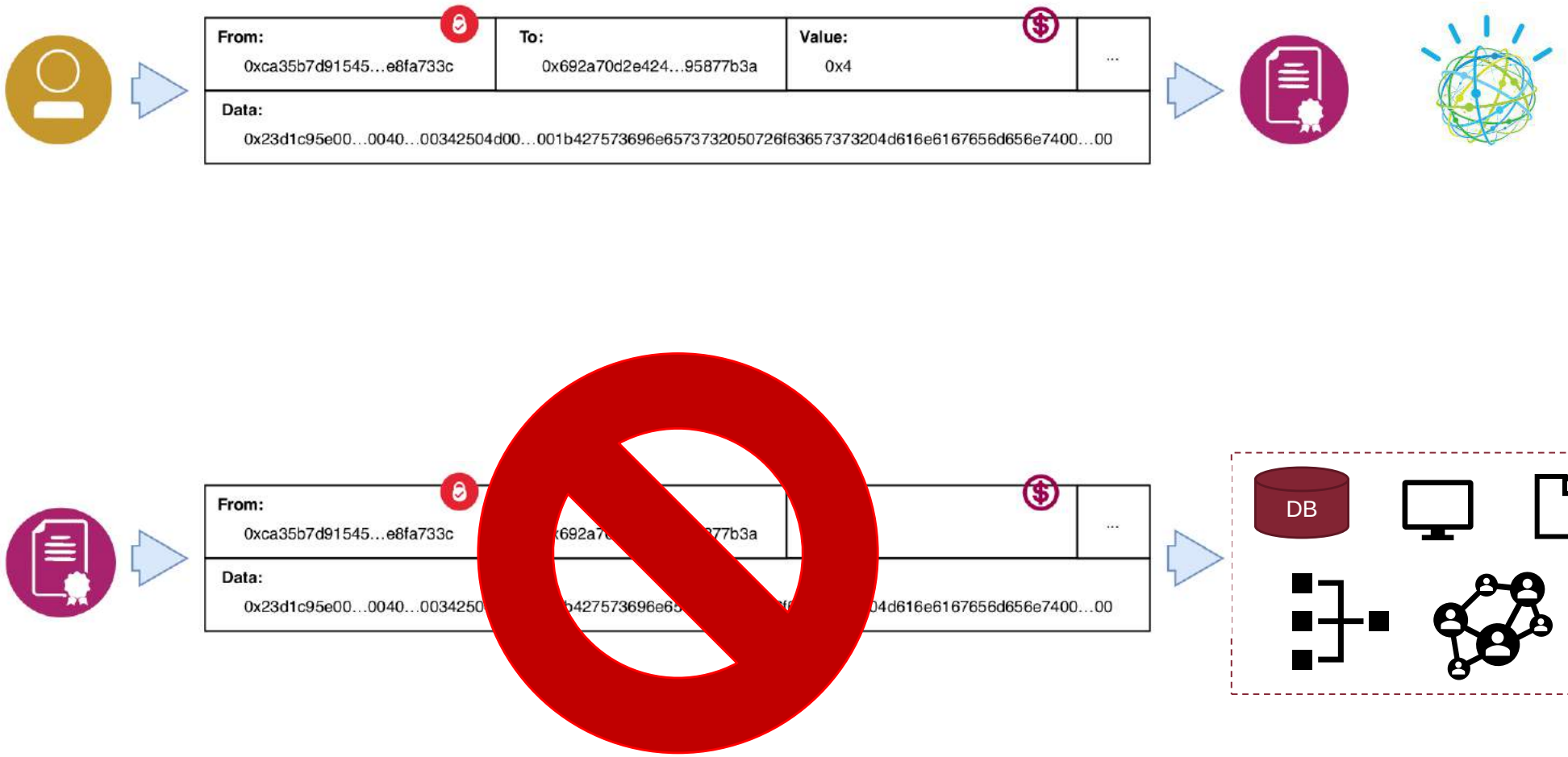
Contact with the off-chain world

Payout in case of signalled problems with the flight



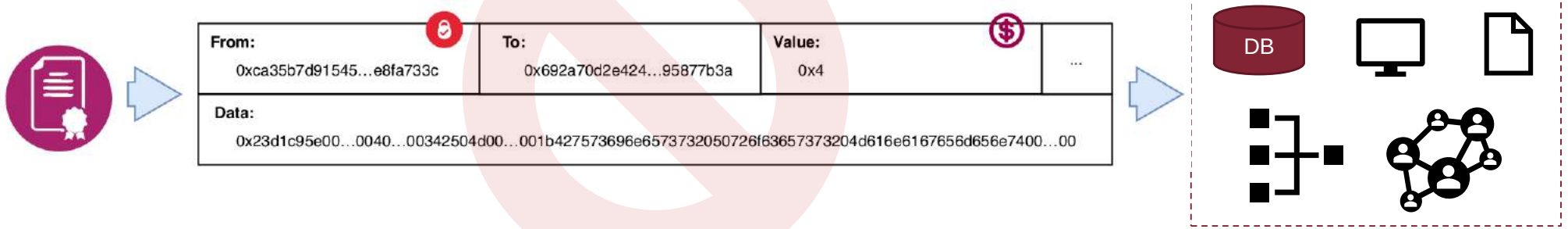
Source: https://www.flickr.com/photos/michaelduxbury/5824469025

The problem


From:
0xca35b7d91545...e8fa733c
To:
0x692a70d2e424...95877b3a
Value:
0x4
...
Data:
0x23d1c95e00...0040...00342504d00...001b427573696e6573732050726f63657373204d616e6167656d656e7400...00
From:
0xca35b7d91545...e8fa733c
...
Data:
DB

The Oracle

Source: http://matrix.wikia.com/wiki/File:The_Oracle_Making_Cookies.jpg

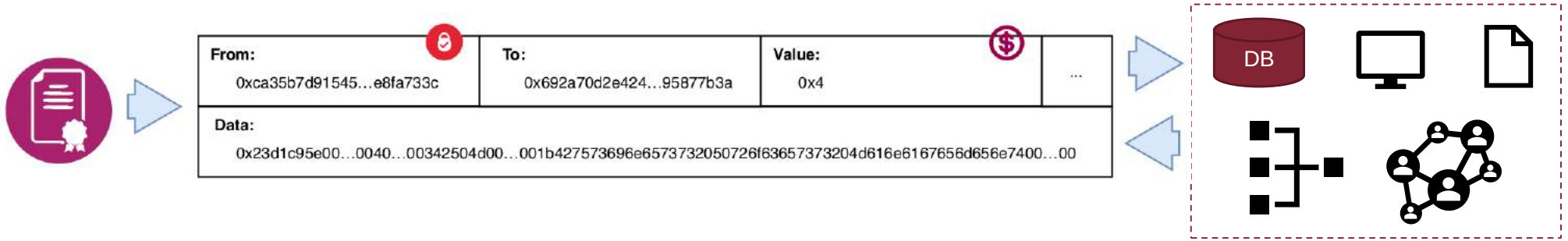


The Oracle

ISO/TC 307, ISO/TR 2345: “[A] **DLT Oracle** [is a] **service** that updates a distributed ledger using **data from outside** the distributed ledger system”. (2019)

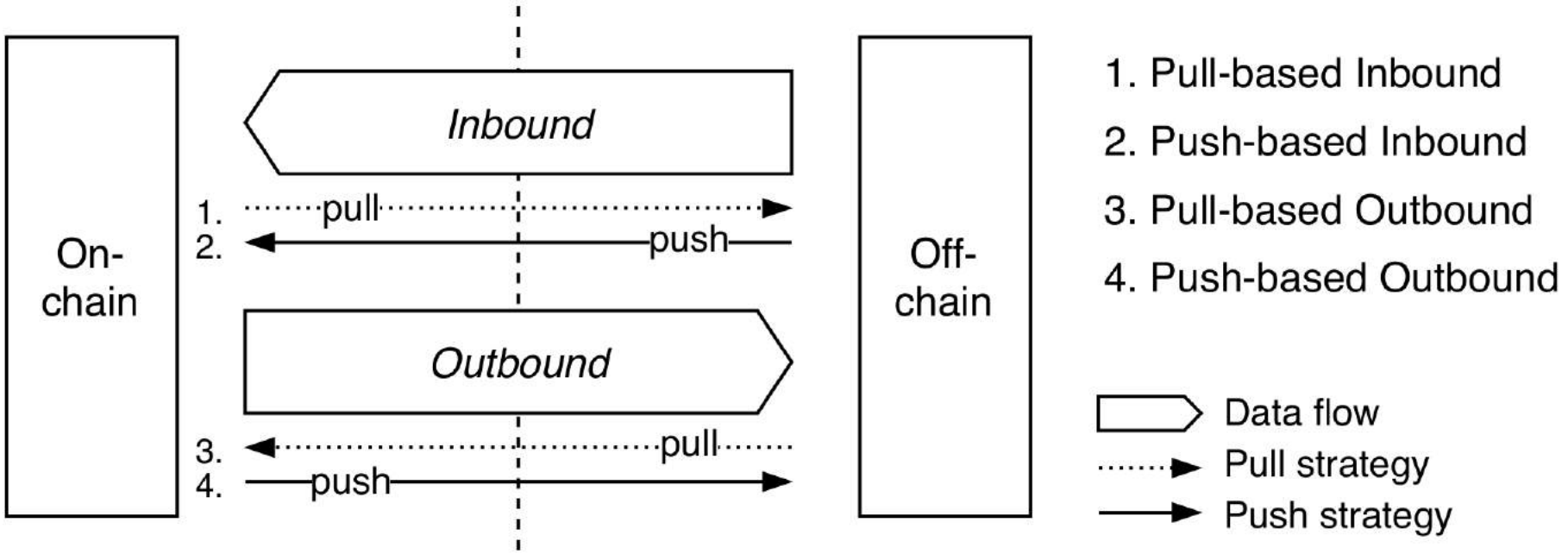
Previous literature: oracles as off-chain information providers.

We see **oracles** as a **bridge**
between the on-chain and off-chain worlds.



Foundational Oracle Patterns: Connecting Blockchain to the Off-Chain World. BPM (Blockchain and RPA Forum) 2020: 35-51

Oracle patterns: Overview



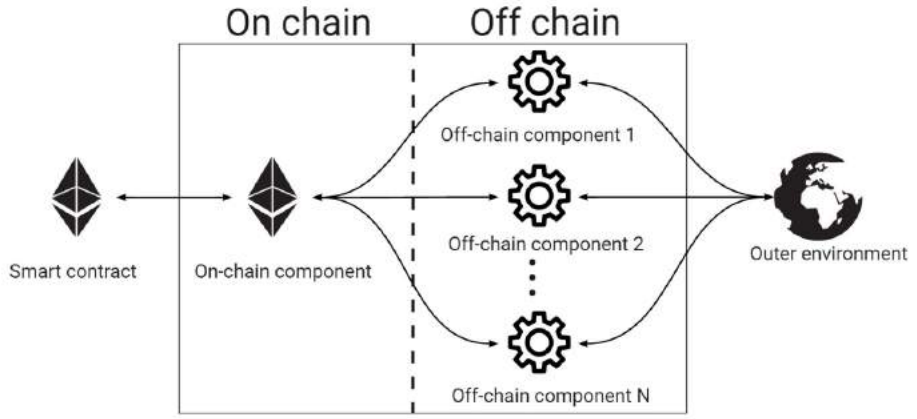
Foundational Oracle Patterns: Connecting Blockchain to the Off-Chain World. BPM (Blockchain and RPA Forum) 2020: 35-51

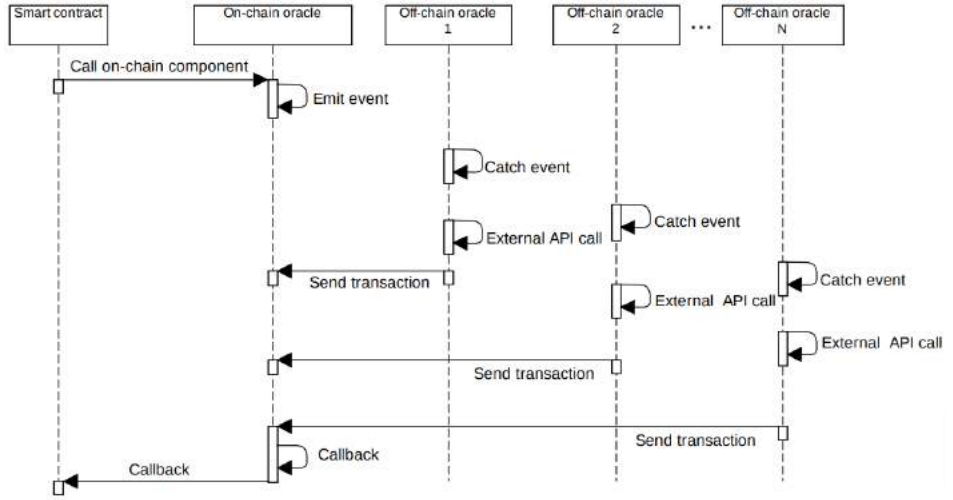
Decentralised oracles

Enhancing Blockchain-Based Processes with Decentralized Oracles

Davide Basile[1], Valerio Goretti[1], Claudio Di Ciccio[1(✉)], and Sabrina Kirrane[2]

[1] Sapienza University of Rome, Rome, Italy
{basile.1810355,goretti.1811110}@studenti.uniroma1.it,
claudio.diciccio@uniroma1.it
[2] Vienna University of Economics and Business, Vienna, Austria
sabrina.kirrane@wu.ac.at

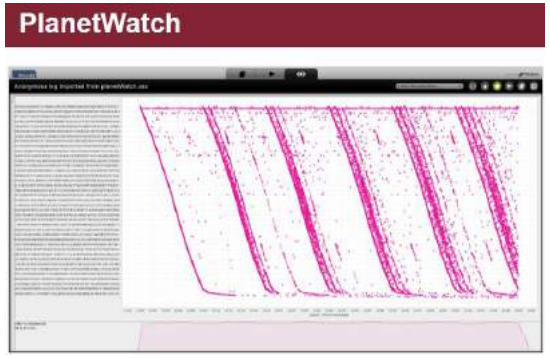


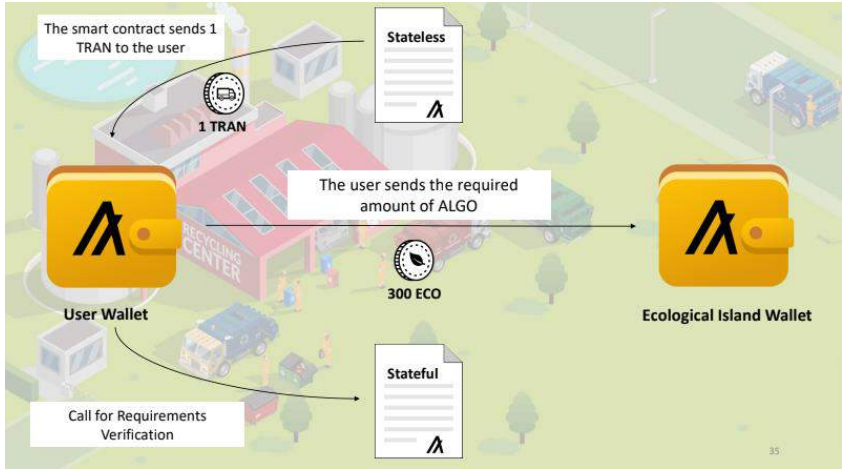


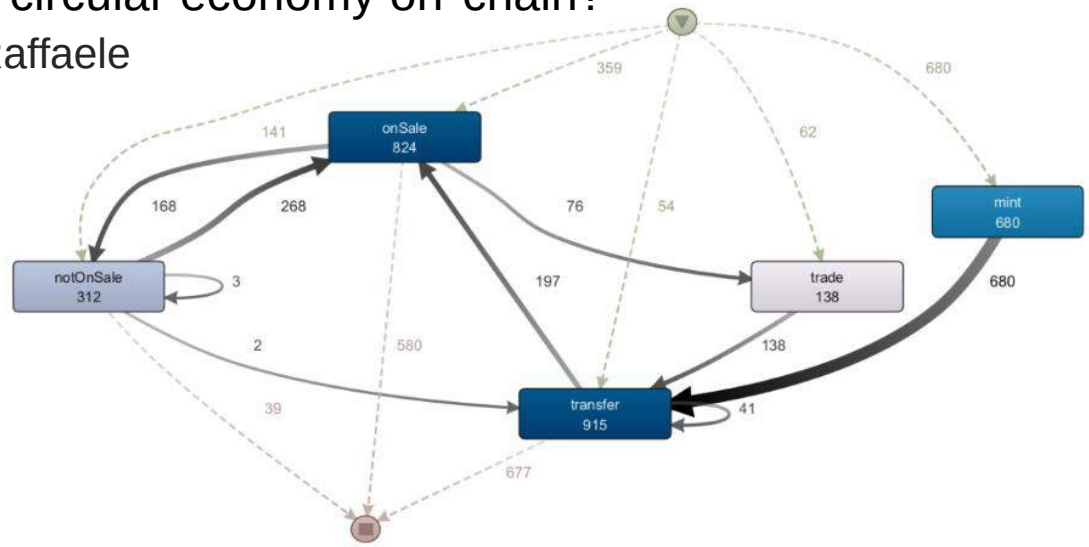
Enhancing Blockchain-Based Processes with Decentralized Oracles. In: BPM Blockchain and RPA Forum 2021, 102-118, Springer.

Currently work-in-progress

- How to restrict information access with data on public blockchains?
  - Ask Edoardo Marangone
- How to extract event logs from Ethereum Classic and Algorand?
  - Ask Michele Kryston and Silverio Manganaro
- How to run declarative process specifications on Algorand?
  - Ask Mirko Politi
- How to design and implement processes for circular economy on-chain?
  - Ask Davide Basile, Valerio Goretti and Marco Raffaele







Images courtesy of the authors. Thanks guys!

SAPIENZA
UNIVERSITÀ DI ROMA

Blockchain as a process execution infrastructure

Claudio Di Ciccio | diciccio.net | claudio.diciccio@uniroma1.it
Assistant Professor, Sapienza University of Rome, Italy | uniroma1.it