Trusted Deployer:

A Tool for Safe Creation and Upgrade of Ethereum Smart Contracts

Juliandson Ferreira[1], Pedro Antonino[2], Augusto Sampaio[1], A. W. Roscoe[2,3] and Filipe Arruda[1]

[1] Universidade Federal de Pernambuco
[2] The Blockhouse Technology Limited
[3] University College Blockchain Research Centre

27th Brazilian Symposium on Formal Methods

Smart contracts

- Smart contracts are programs stored on a blockchain that automatically enforce its terms when predetermined conditions are met
- They eliminate the need for intermediaries by enforcing agreements between parties
- They were created to provide a secure way to manage digital assets

Code is law

- Building blocks: smart contracts
- Code is **immutable** and **autonomous**
- Code unequivocally and unambiguously defines behaviour

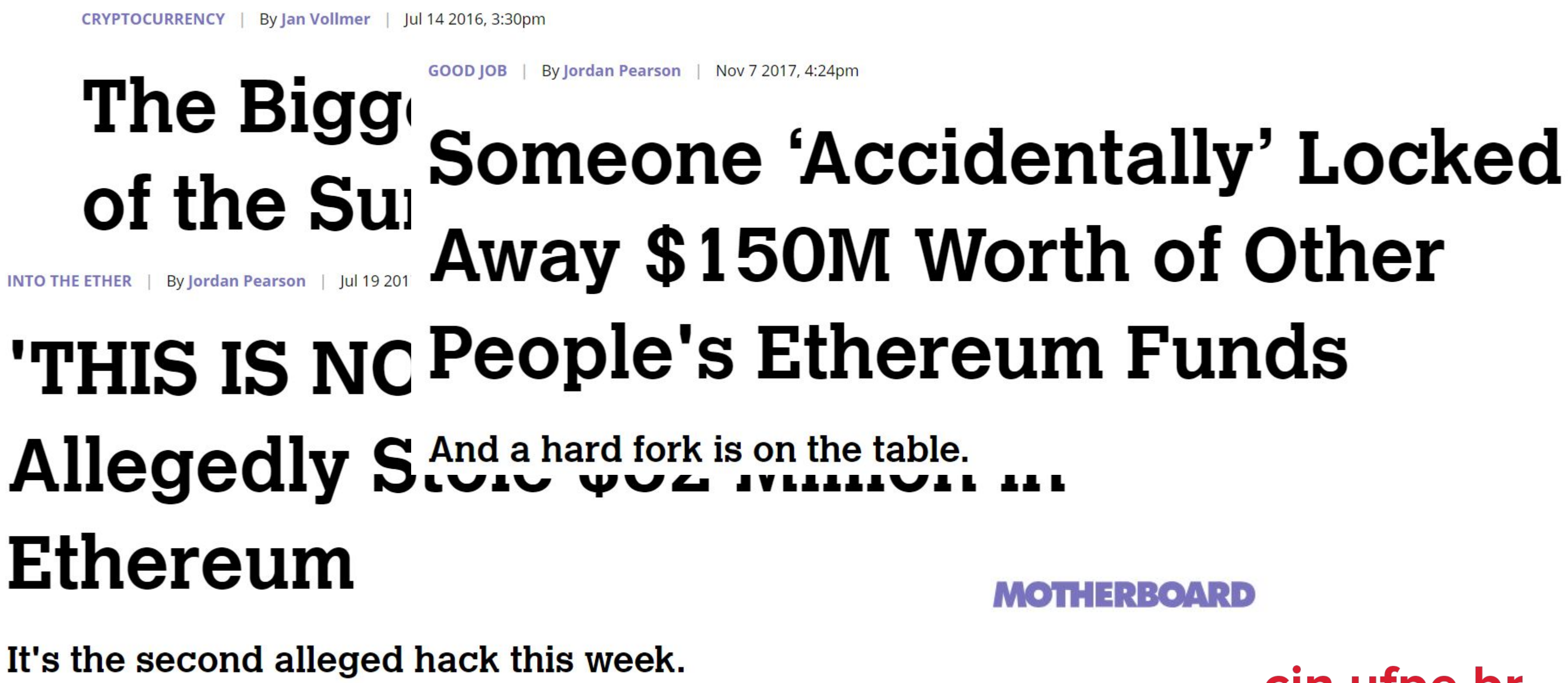
Code is law

- Building blocks: smart contracts
- Code is **immutable**
- Code unequivocally and unambiguously defines behaviour

What if the code is wrong?

Centro de Informática UFPE | Universidade Federal de Pernambuco

Attacks on smart contracts



CRYPTOCURRENCY | By Jan Vollmer | Jul 14 2016, 3:30pm

The Bigg
of the Su

GOOD JOB | By Jordan Pearson | Nov 7 2017, 4:24pm

Someone 'Accidentally' Locked Away $150M Worth of Other People's Ethereum Funds

And a hard fork is on the table.

INTO THE ETHER | By Jordan Pearson | Jul 19 201

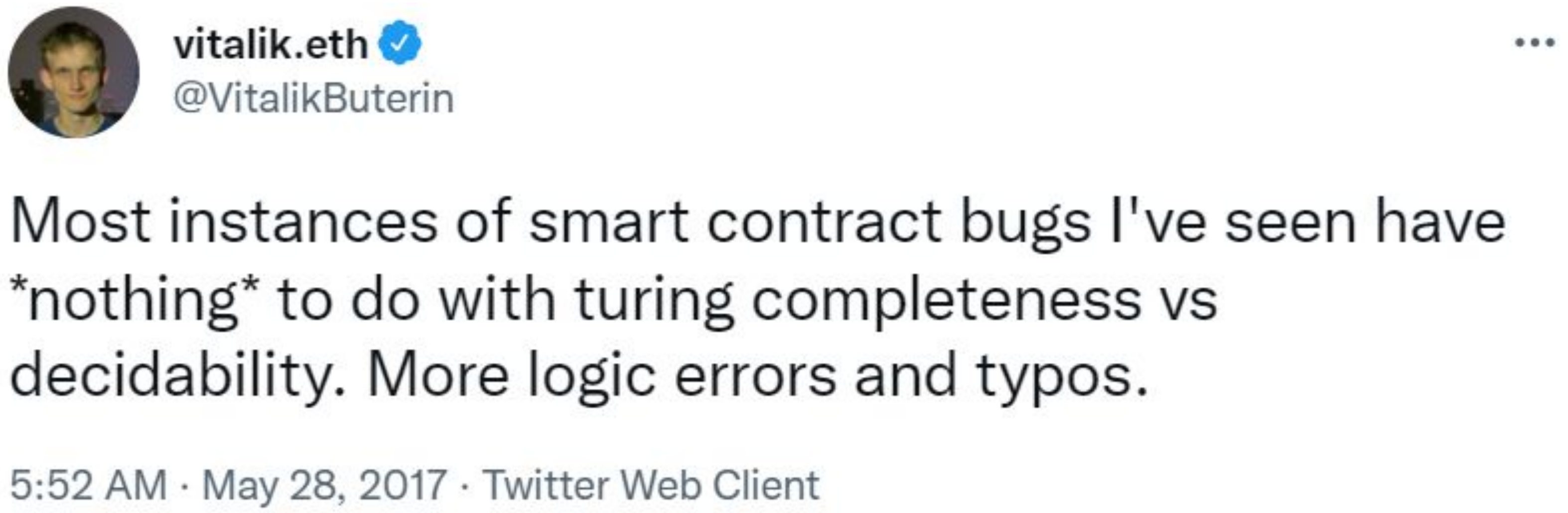
'THIS IS NO
Allegedly S
Ethereum

It's the second alleged hack this week.

MOTHERBOARD

cin.ufpe.br

Problem



vitalik.eth

@VitalikButerin

Most instances of smart contract bugs I've seen have *nothing* to do with turing completeness vs decidability. More logic errors and typos.

5:52 AM · May 28, 2017 · Twitter Web Client

This tweet can be found in: https://twitter.com/vitalikbuterin/status/868751724311216128

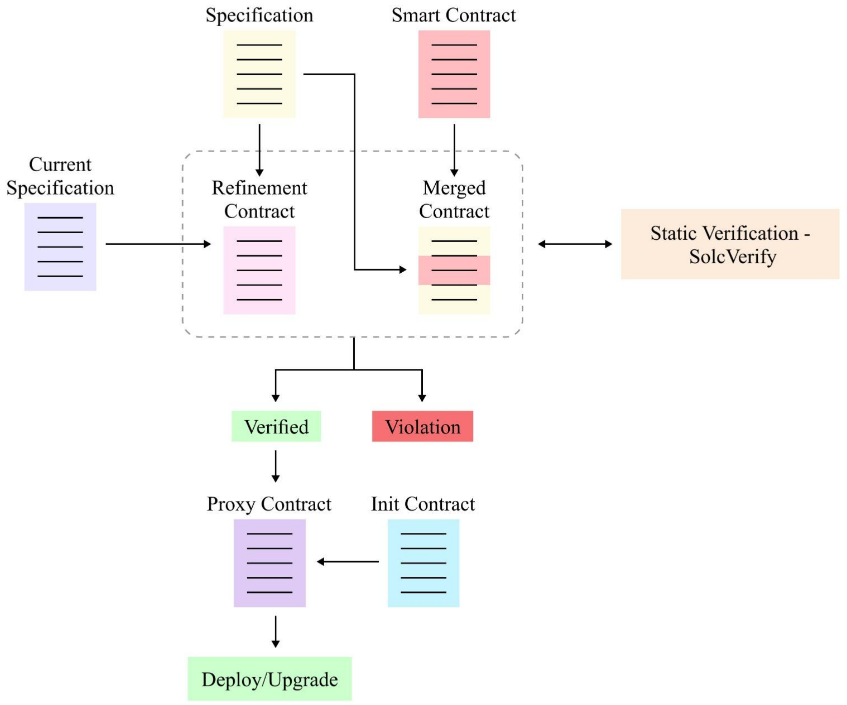
State of the art

- A number of tools to analyse smart contracts
  - Try to prevent bugs
- The proxy pattern
  - Allow simulation of contract upgrades
- Contract auditing
  - Manual/tool-supported detailed code reviews

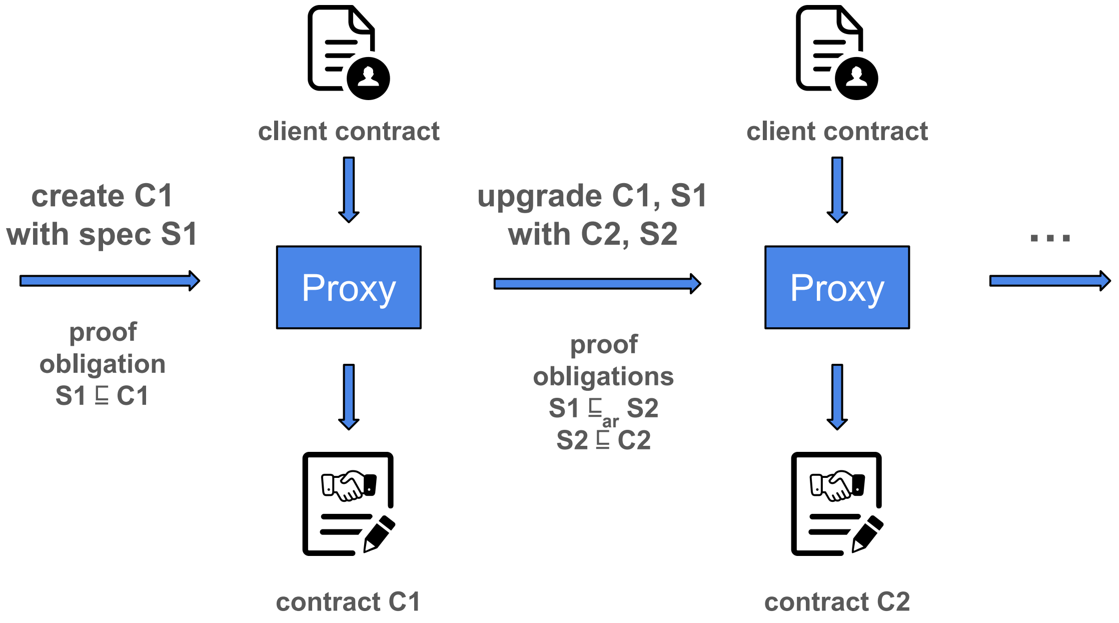
State of the art

- A number of tools to analyse smart contracts
    - Try to prevent bugs **--- no systematic application/enforcement framework**
- The proxy pattern
    - Allow simulation of contract upgrades **--- no upgrade guarantees/too late**
- Contract auditing
    - Manual/tool-supported detailed code reviews **--- no formal guarantees**

Tool support and application - trusted deployer (Proposal)



A typical safe evolution scenario

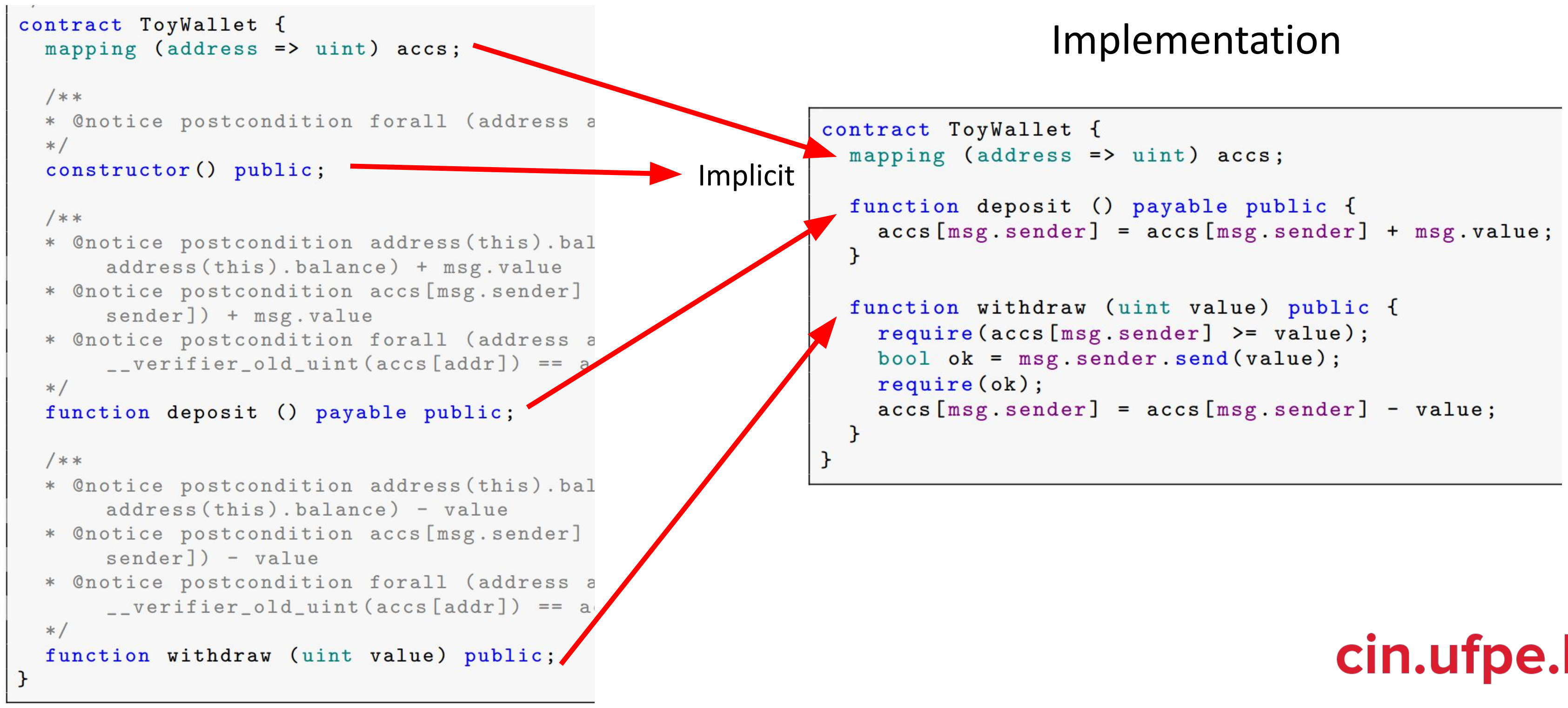


Paradigm shift: from ***code is law*** to ***conformance is law***

Conformance notion: syntactic obligation

Specification

Implementation

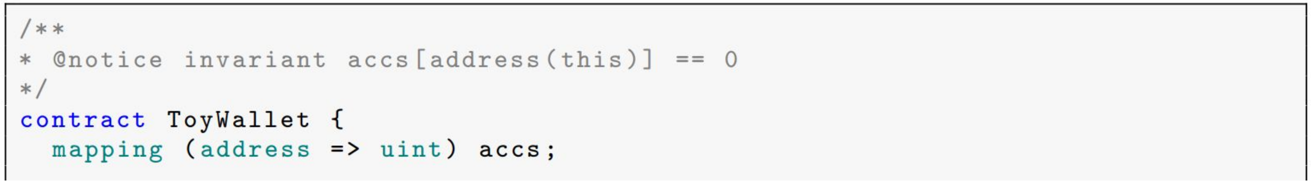


```
contract ToyWallet {
  mapping (address => uint) accs;

  /**
  * @notice postcondition forall (address a
  */
  constructor() public;

  /**
  * @notice postcondition address(this).bal
        address(this).balance) + msg.value
  * @notice postcondition accs[msg.sender]
        sender]) + msg.value
  * @notice postcondition forall (address a
        __verifier_old_uint(accs[addr]) == a
  */
  function deposit () payable public;

  /**
  * @notice postcondition address(this).bal
        address(this).balance) - value
  * @notice postcondition accs[msg.sender]
        sender]) - value
  * @notice postcondition forall (address a
        __verifier_old_uint(accs[addr]) == a
  */
  function withdraw (uint value) public;
}
```

```
contract ToyWallet {
  mapping (address => uint) accs;

  function deposit () payable public {
    accs[msg.sender] = accs[msg.sender] + msg.value;
  }

  function withdraw (uint value) public {
    require(accs[msg.sender] >= value);
    bool ok = msg.sender.send(value);
    require(ok);
    accs[msg.sender] = accs[msg.sender] - value;
  }
}
```

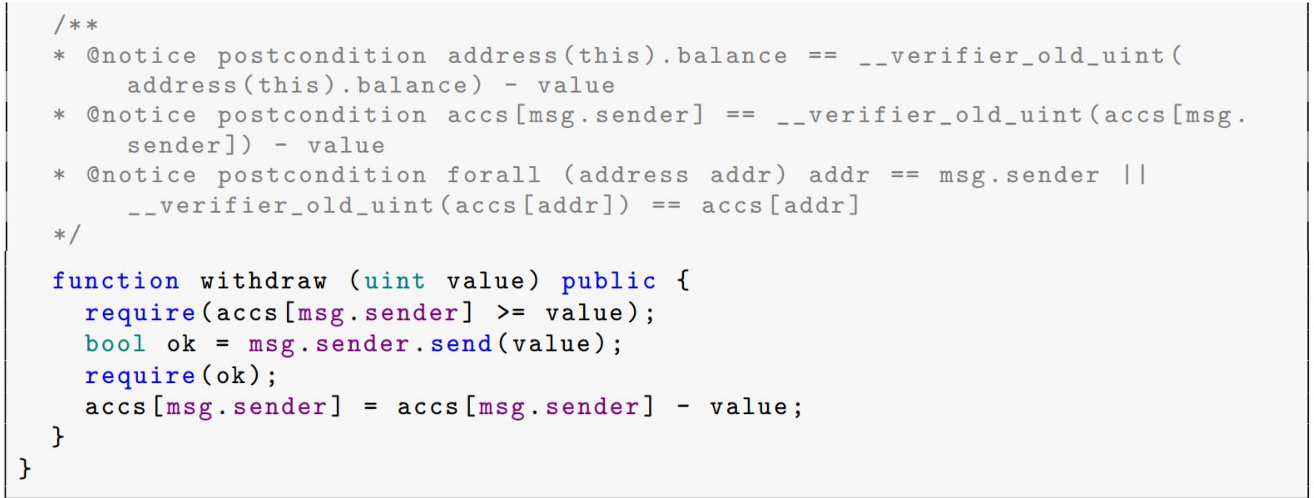
Conformance notion: semantic obligation

Merged contract



```
/**
* @notice invariant accs[address(this)] == 0
*/
contract ToyWallet {
  mapping (address => uint) accs;
```

… constructor and deposit omitted…

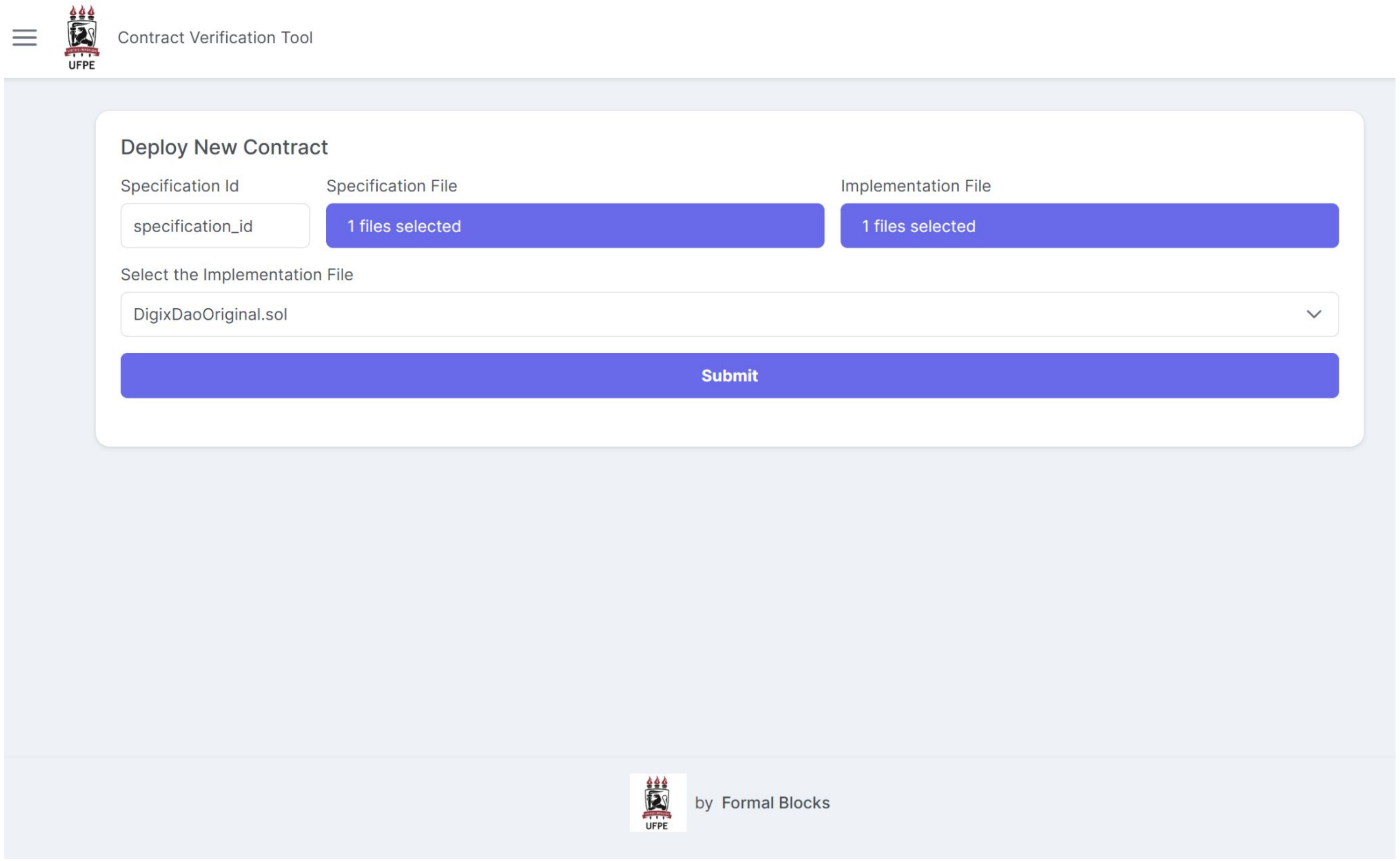


```
  /**
  * @notice postcondition address(this).balance == __verifier_old_uint(
      address(this).balance) - value
  * @notice postcondition accs[msg.sender] == __verifier_old_uint(accs[msg.
      sender]) - value
  * @notice postcondition forall (address addr) addr == msg.sender ||
      __verifier_old_uint(accs[addr]) == accs[addr]
  */

  function withdraw (uint value) public {
    require(accs[msg.sender] >= value);
    bool ok = msg.sender.send(value);
    require(ok);
    accs[msg.sender] = accs[msg.sender] - value;
  }
}
```

solc-verify

- off-the-shelf verifier
- design by contract

Safe contract creation



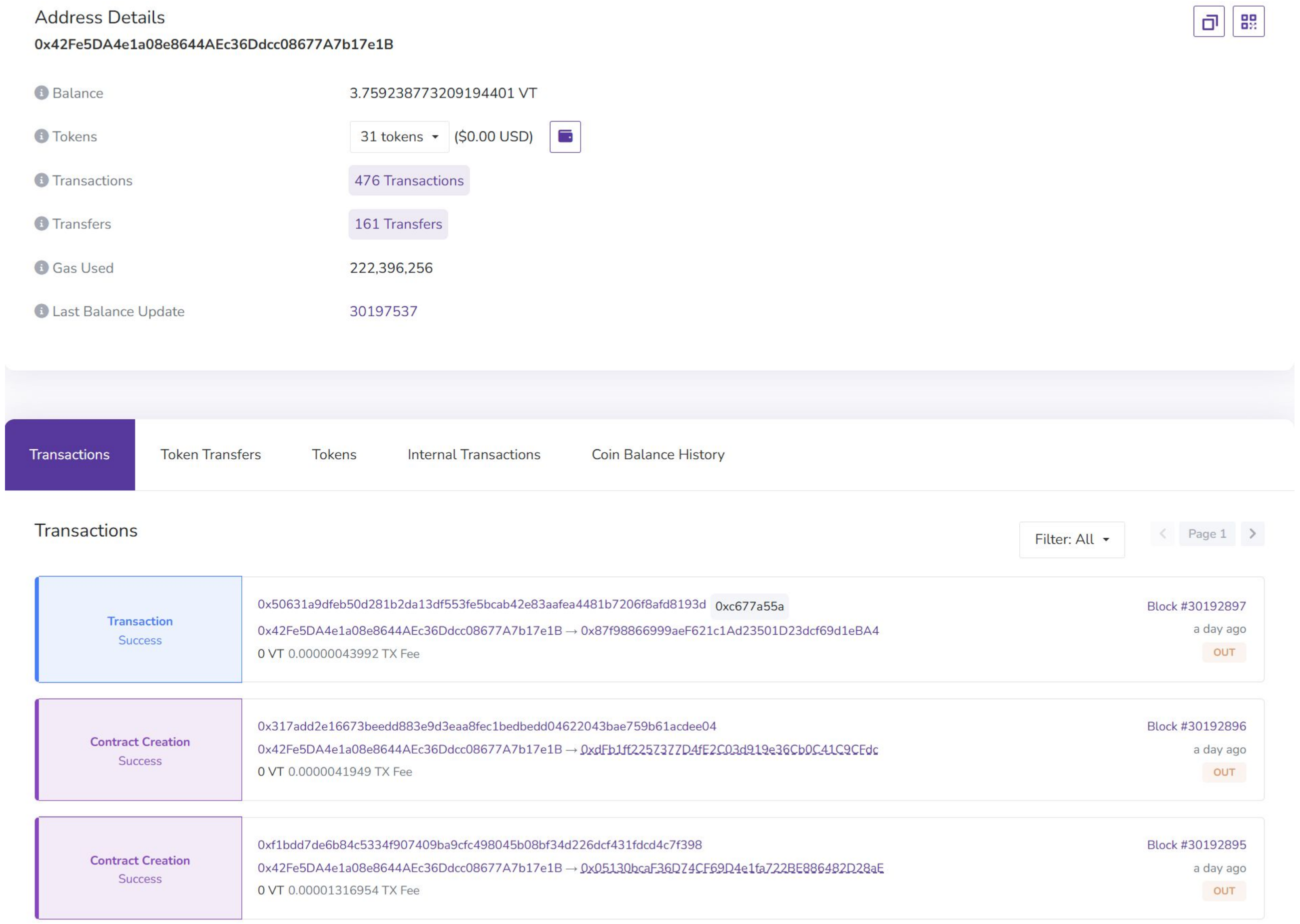
Contract Verification Tool

Deploy New Contract

Specification Id	Specification File	Implementation File
specification_id	1 files selected	1 files selected

Select the Implementation File: DigixDaoOriginal.sol

Safe contract creation transactions



Address Details
0x42Fe5DA4e1a08e8644AEc36Ddcc08677A7b17e1B

Balance	3.759238773209194401 VT
Tokens	31 tokens ($0.00 USD)
Transactions	476 Transactions
Transfers	161 Transfers
Gas Used	222,396,256
Last Balance Update	30197537

Transactions | Token Transfers | Tokens | Internal Transactions | Coin Balance History

Transactions

Filter: All — Page 1

Transaction
Success
0x50631a9dfeb50d281b2da13df553fe5bcab42e83aafea4481b7206f8afd8193d 0xc677a55a
0x42Fe5DA4e1a08e8644AEc36Ddcc08677A7b17e1B → 0x87f98866999aeF621c1Ad23501D23dcf69d1eBA4
0 VT 0.00000043992 TX Fee
Block #30192897
a day ago
OUT

Contract Creation
Success
0x317add2e16673beedd883e9d3eaa8fec1bedbedd04622043bae759b61acdee04
0x42Fe5DA4e1a08e8644AEc36Ddcc08677A7b17e1B → 0xdFb1ff2257377D4fE2C03d919e36Cb0C41C9CFdc
0 VT 0.0000041949 TX Fee
Block #30192896
a day ago
OUT

Contract Creation
Success
0xf1bdd7de6b84c5334f907409ba9cfc498045b08bf34d226dcf431fdcd4c7f398
0x42Fe5DA4e1a08e8644AEc36Ddcc08677A7b17e1B → 0x05130bcaF36D74CF69D4e1fa722BE886482D28aE
0 VT 0.00001316954 TX Fee
Block #30192895
a day ago
OUT

cin.ufpe.br

Specification Refinement

Centro de Informática UFPE | Universidade Federal de Pernambuco

Original Spec

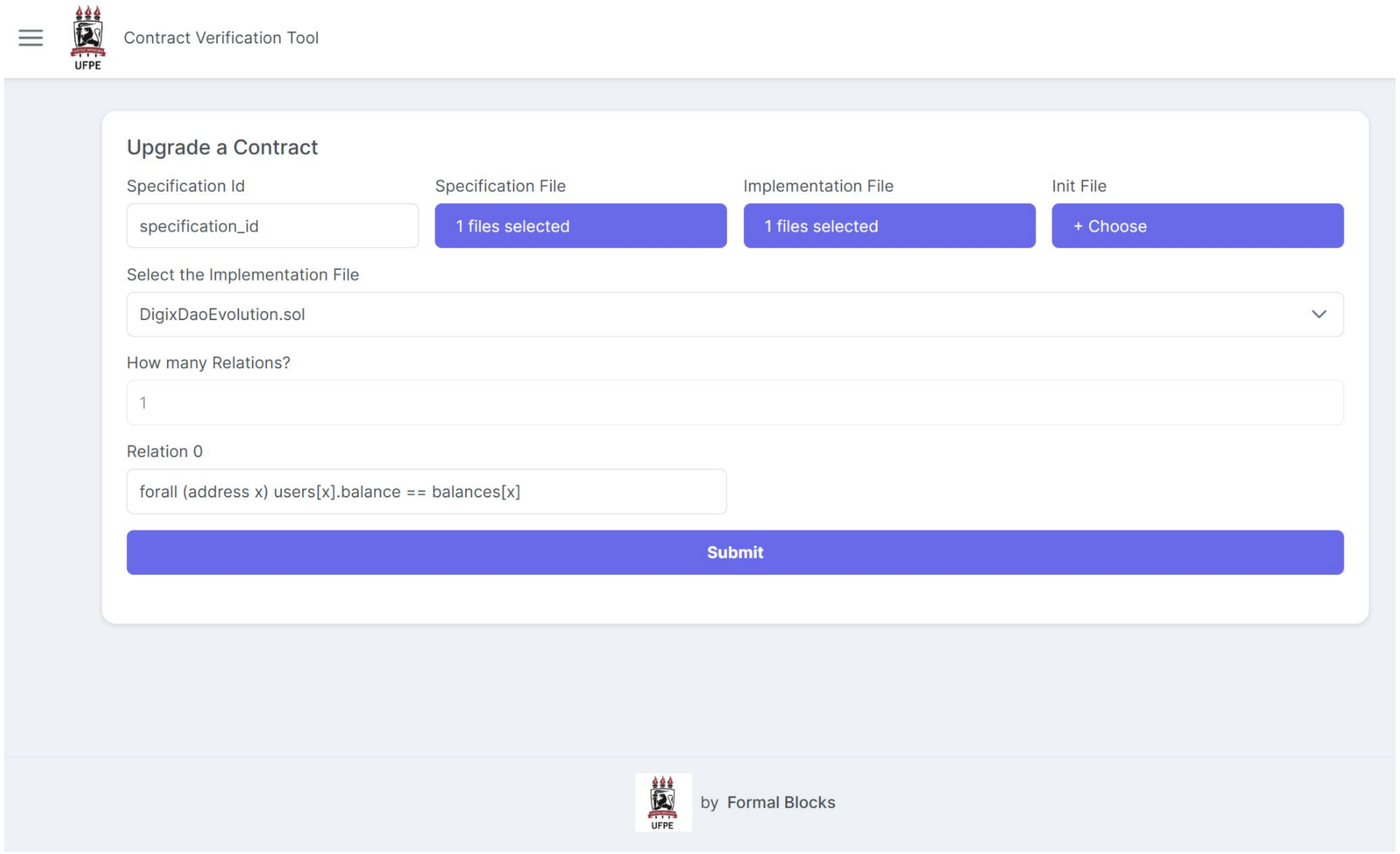
```
//@notice invariant totalSupply == __verifier_sum_uint(users[__verifier_idx_address].balance)
contract ERC20Spec {
  struct User {
    uint256 balance;
  }

  mapping (address => User) users;


  // @notice postcondition  users[_owner].balance == balance
  function balanceOf(address _owner) public returns (uint256 balance);

  /**
  * @notice  postcondition ( ( users[msg.sender].balance ==  __verifier_old_uint
(users[msg.sender].balance ) - _value  && msg.sender  != _to ) ||   ( users[msg.sender].balance
==  __verifier_old_uint ( users[msg.sender].balance ) && msg.sender  == _to ) &&  success )  ||
!success

   * @notice  postcondition ( ( users[_to].balance ==  __verifier_old_uint ( users[_to].balance
) + _value  && msg.sender  != _to ) ||   ( users[_to].balance ==  __verifier_old_uint (
users[_to].balance ) && msg.sender  == _to ) &&  success )   || !success
   */
  function transfer(address _to, uint256 _value) public returns (bool success);
}
```

Refined Spec

```
// @notice invariant  totalSupply  ==  __verifier_sum_uint(balances)
contract ERC20SpecRefined {

  mapping (address => uint256) balances;



  /// @notice postcondition  balances[_owner] == balance
  function balanceOf(address _owner) public returns (uint256 balance);


  /**
   * @notice  postcondition ( ( balances[msg.sender] ==  __verifier_old_uint
(balances[msg.sender] ) - _value  && msg.sender  != _to ) ||   (
balances[msg.sender] ==  __verifier_old_uint ( balances[msg.sender] ) &&
msg.sender  == _to ) &&  success )  || !success

  * @notice  postcondition ( ( balances[_to] ==  __verifier_old_uint (
balances[_to] ) + _value  && msg.sender  != _to ) ||   ( balances[_to] ==
__verifier_old_uint ( balances[_to] ) && msg.sender  == _to ) &&  success )
|| !success
  */
  function transfer(address _to, uint256 _value) public returns (bool success);
}
```

Abstraction relation: **forall (address a) users[a].balance == balances[a]**

Safe contract upgrade



Contract Verification Tool

Upgrade a Contract

Specification Id: specification_id

Specification File: 1 files selected

Implementation File: 1 files selected

Init File: + Choose

Select the Implementation File: DigixDaoEvolution.sol

How many Relations? 1

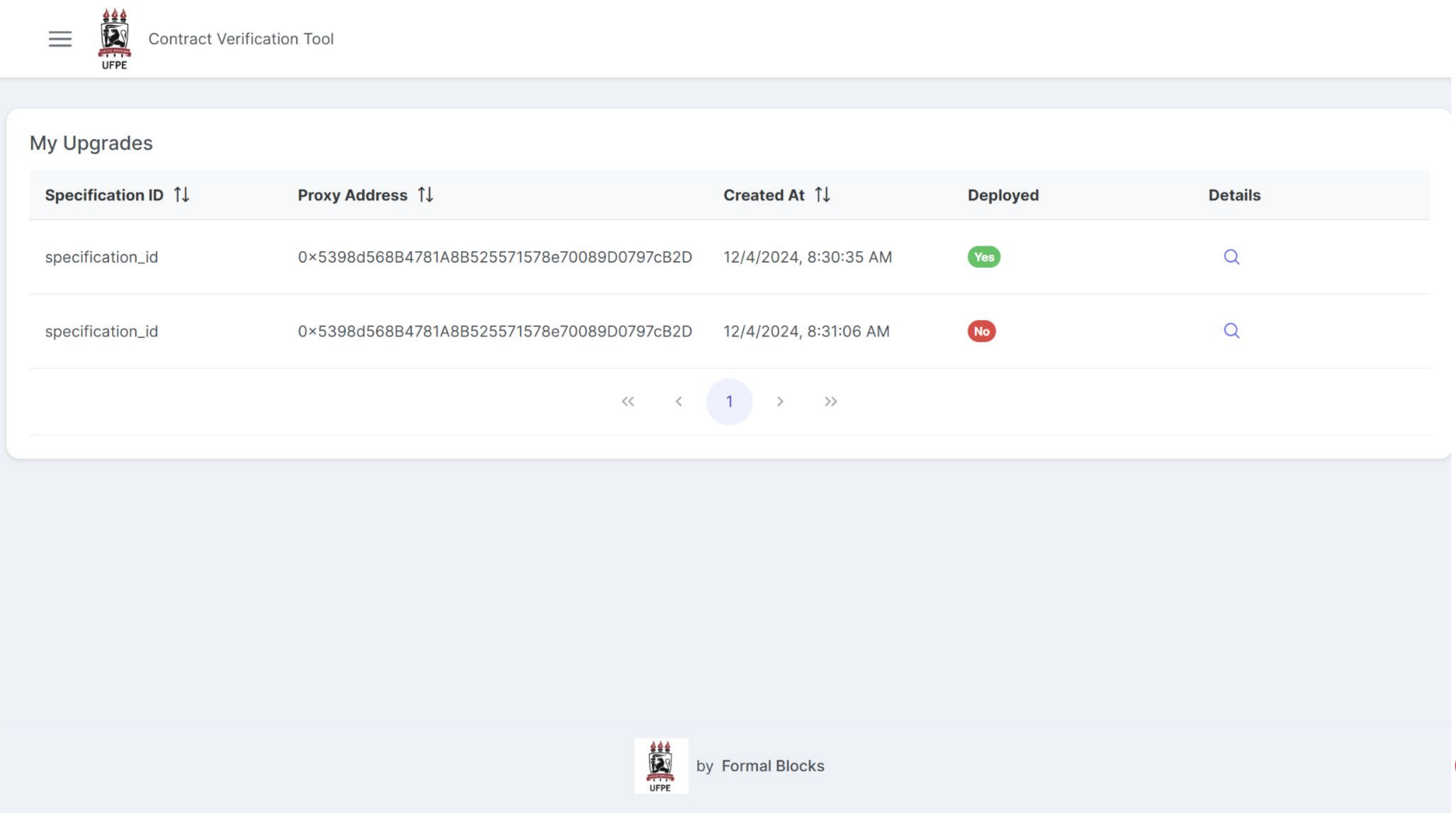
Relation 0: forall (address x) users[x].balance == balances[x]

by Formal Blocks

My Upgrades

Centro de Informática UFPE

Universidade Federal de Pernambuco



My Upgrades

Specification ID	Proxy Address	Created At	Deployed	Details
specification_id	0x5398d568B4781A8B525571578e70089D0797cB2D	12/4/2024, 8:30:35 AM	Yes	🔍
specification_id	0x5398d568B4781A8B525571578e70089D0797cB2D	12/4/2024, 8:31:06 AM	No	🔍

cin.ufpe.br

Background Theory

- Pedro Antonino, **Juliandson Ferreira**, Augusto Sampaio, and A. W. Roscoe. Specification is law: Safe creation and upgrade of ethereum smart contracts. In Bernd-Holger Schlingloff and Ming Chai, editors, Software Engineering and Formal Methods - **20th International Conference, SEFM 2022**, Berlin, Germany, September 26-30, 2022, Proceedings, volume 13550 of Lecture Notes in Computer Science, pages 227–243. Springer, 2022.

- Pedro Antonino, **Juliandson Ferreira**, Augusto Sampaio, and A. W. Roscoe. A refinement-based approach to safe smart contract deployment and evolution. In **Software and Systems Modeling, SOSYM 2024**, page 657–693, Cham, 2024. Springer International Publishing.

Conclusion

- Our framework is centred around a trusted deployer that prevents the creation and upgrade of non-compliant contracts.
- Trusted deployer records information about the contracts that have been verified, and which specification they conform to.
- Evaluation Ethereum Standards: ERC20, ERC3156, ERC721 and ERC1155.

Future Work

- Systematic mapping from informal requirements to formal specifications
- Investigate bugs arising from the consensus protocol
- Automate the migration of the contract state when the upgrade involves a change in data representation

Trusted Deployer:

A Tool for Safe Creation and Upgrade of Ethereum Smart Contracts

Juliandson Ferreira[1], Pedro Antonino[2], Augusto Sampaio[1], A. W. Roscoe[2,3] and Filipe Arruda[1]

[1] Universidade Federal de Pernambuco
[2] The Blockhouse Technology Limited
[3] University College Blockchain Research Centre

27th Brazilian Symposium on Formal Methods