NFPs 3: Dependability and Security

Mark Staples
Email: mark.staples@data61.csiro.au

www.data61.csiro.au

Outline

- Dependability and Security
- Trust and the Need for Trustworthy Blockchain Applications
- Functionality
- Security
- Reliability
- Summary
- Assignment 2

Dependability and Security

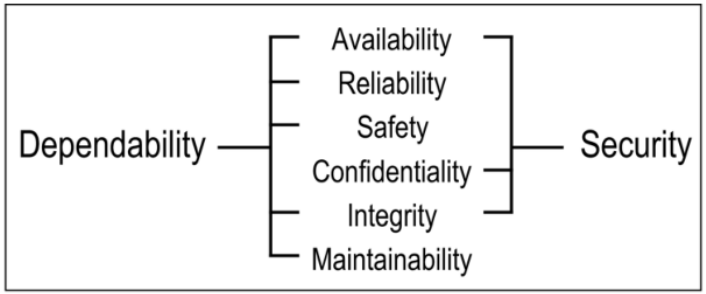


Fig. 1. Dependability and security attributes.

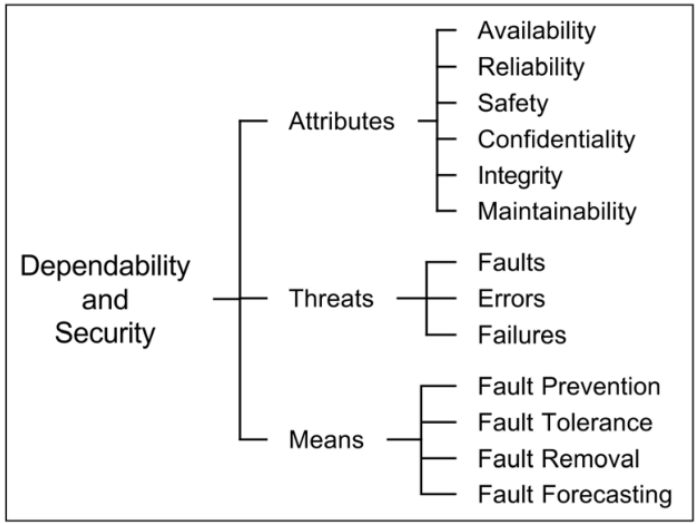


Fig. 2. The dependability and security tree.

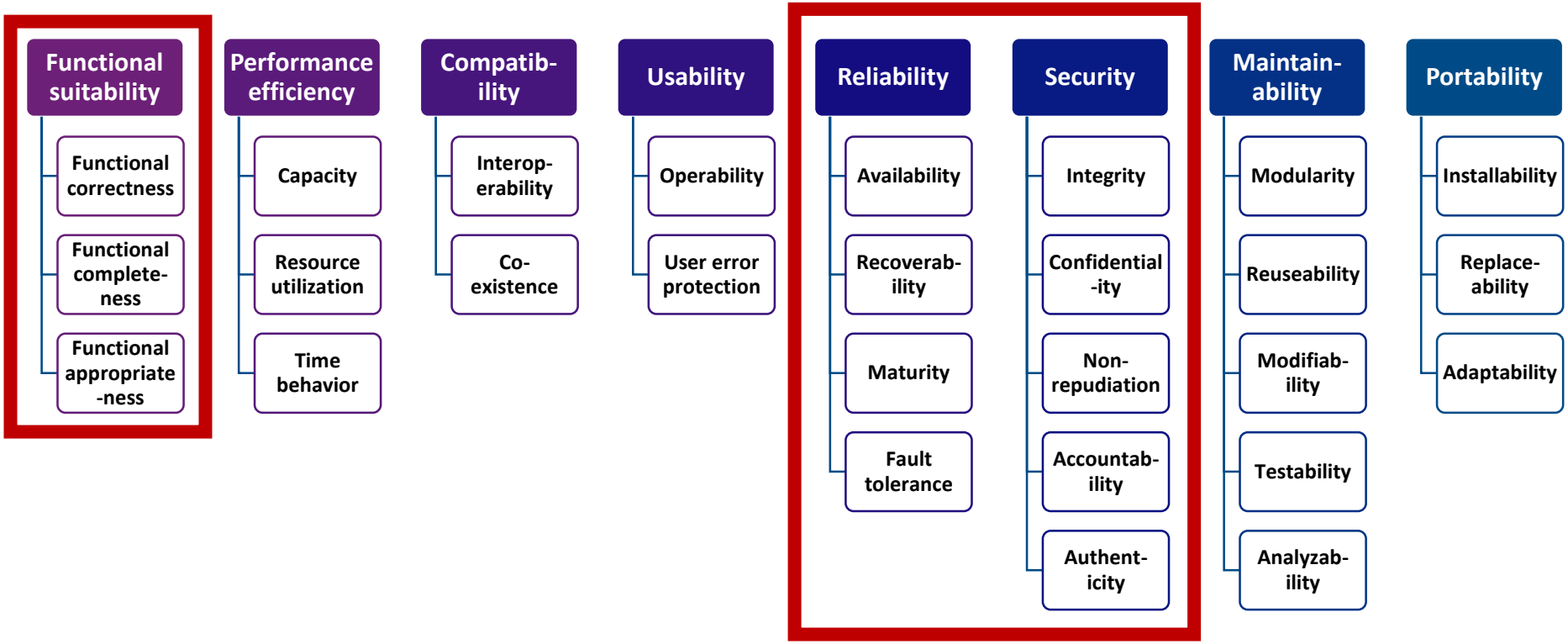
"Basic concepts and taxonomy of dependable and secure computing"
https://www.nasa.gov/pdf/636745main_day_3-algirdas_avizienis.pdf

Faults, Errors, Failures

- **Correct service** is delivered when the service implements the system function.
- A **(service) failure**, is an event that occurs when the delivered service deviates from correct service.
- An **error** is a system state that could lead to a service failure.
- A **fault** is the adjudged or hypothesized cause of an error.

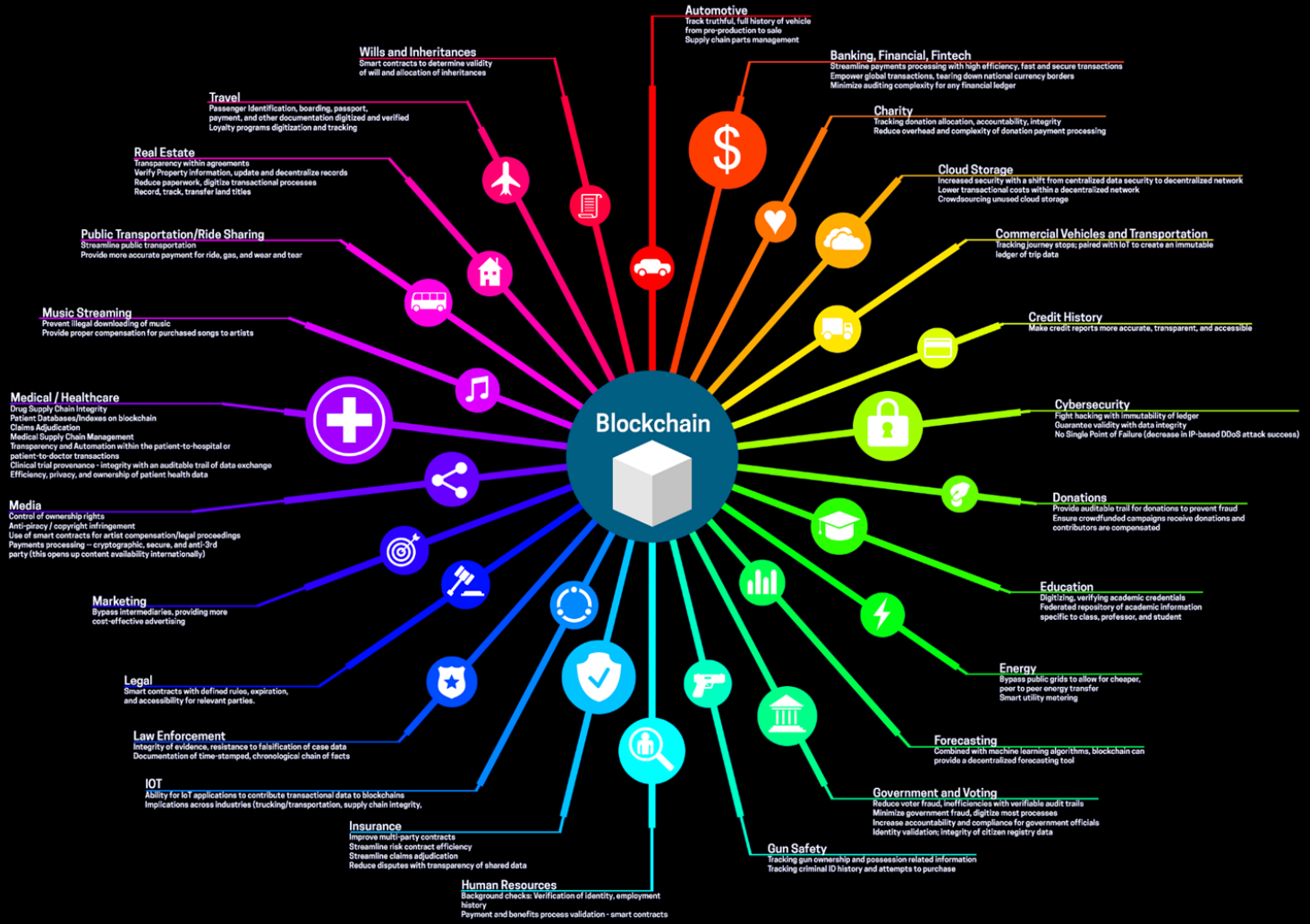
- **Fault prevention** means to prevent the occurrence or introduction of faults.
- **Fault tolerance** means to avoid service failures in the presence of faults.
- **Fault removal** means to reduce the number and severity of faults.
- **Fault forecasting** means to estimate the present number, the future incidence, and the likely consequences of faults.

ISO/IEC 25010:2011 Quality Model



Trust, and the Need for Trustworthy Blockchain Applications

CSIRO DATA 61



https://medium.com/fluree/blockchain-for-2018-and-beyond-a-growing-list-of-blockchain-use-cases-37db7c19fb99

We (Will) Rely on Blockchain-Based Systems

- Smart contract bugs: DAO ($60M); Parity ($280M)
- Cryptographic key loss
  - Hacking: Mt Gox ($450M), Bitfinex ($72M), total Jan-Sep 2018 ($927M); UK rubbish tip ($146M); guns e.g. NYC ($1.8M)

- Huge future economic value (the main point!)
  - e.g. supply chain, asset registries, settlement, energy, …

- Security-critical and Safety-critical use cases
  - e.g. e-health records, food safety, pharma supply chain, IoT management, cybersecurity, law enforcement, …

About “Trust”

- Dependable Software Systems says…
  - Trusted System
    - A system you have chosen to rely on to fulfil a goal
    - When it fails, you suffer harm or loss
  - Trustworthy System
    - A system where you have evidence it will not fail

- “Trustless” Blockchains? **MYTH!**
  - Can shift trust within a wider system
  - Blockchains themselves become trusted components
  - Other components are also trusted
    - e.g. Oracles, Key Management Systems, …

Engineering: Assurance is Key

- Distinguishes engineering from e.g. architecture, building

- Assurance is…
  - A claim that an **artefact** meets key **requirements**
  - Made & evaluated by recognised experts
  - On the basis of a rationale

加工的

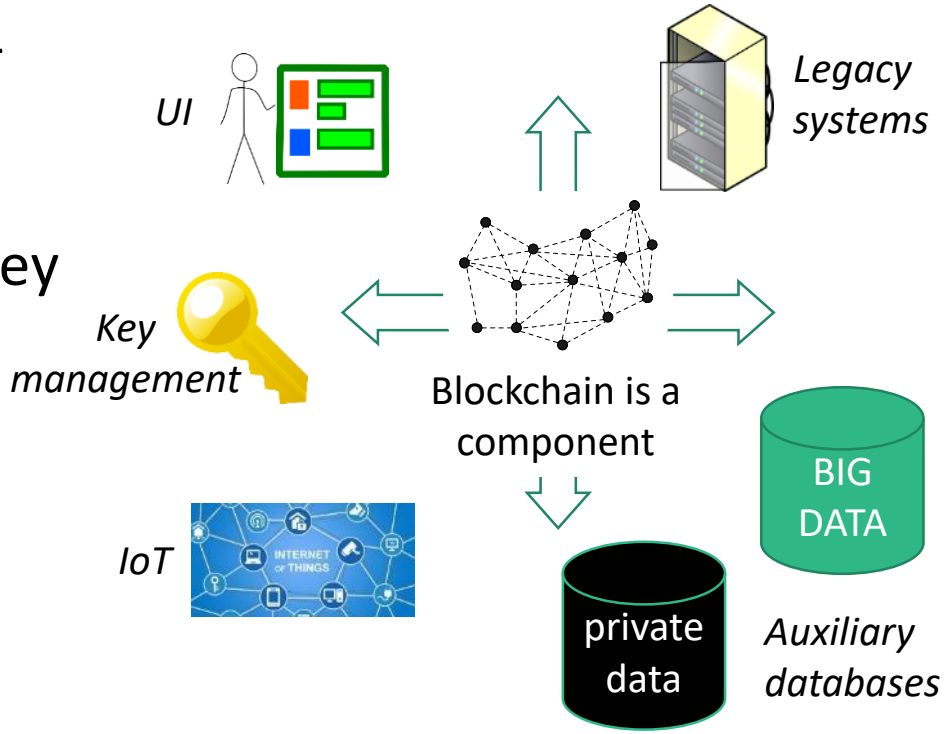
逻辑依据

- Assurance rationale must be backed by empirical **engineering theories** & evidence

以观察和实验为依据的

Trustworthy Blockchain-Based Systems?

- How do we design blockchain-based systems that work?
- What is good evidence that they will work?
  - Functional correctness
  - Non-Functional properties
- How do we get acceptance?
  - Individual, Enterprise, Regulatory, Societal



Functionality

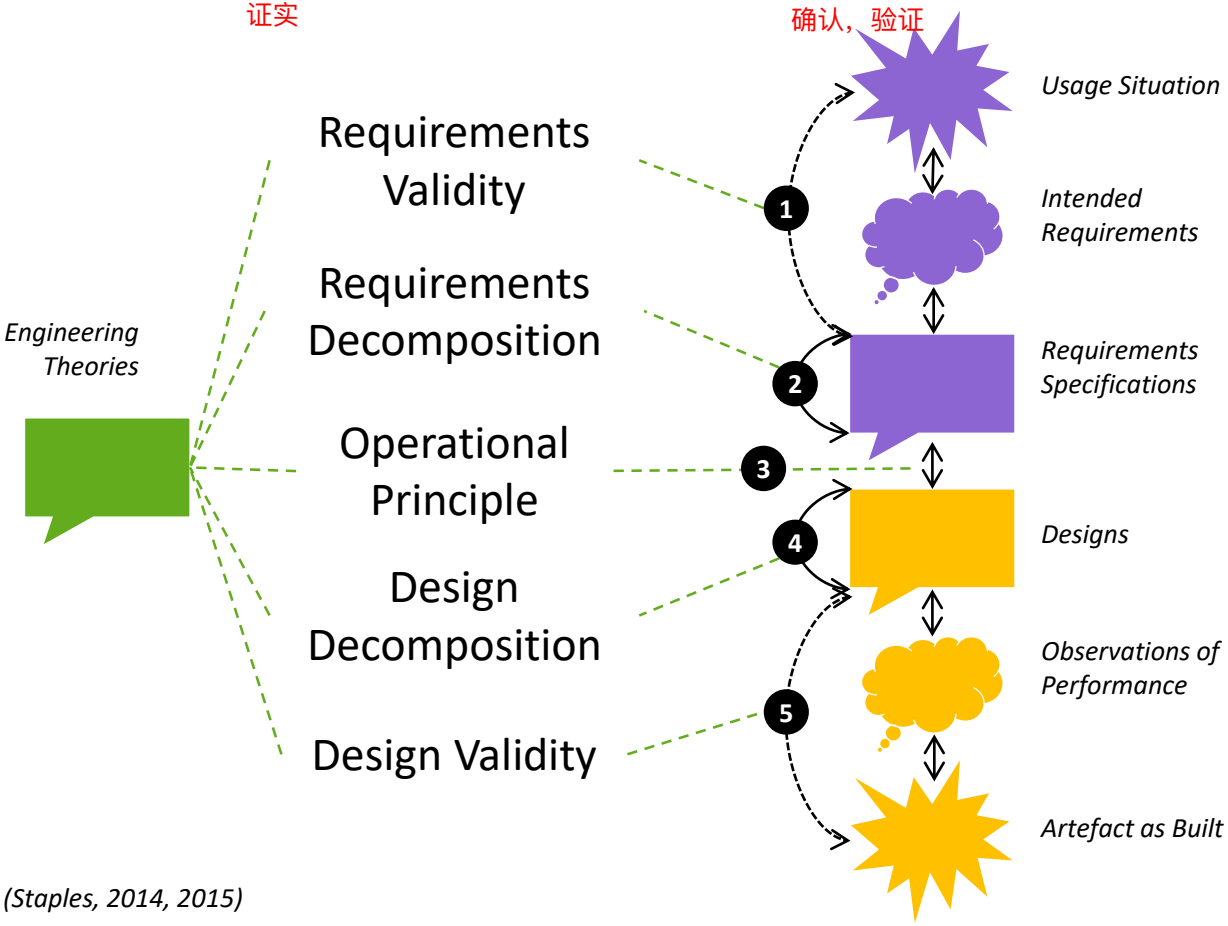
Functional Suitability from ISO/IEC 25010:2011

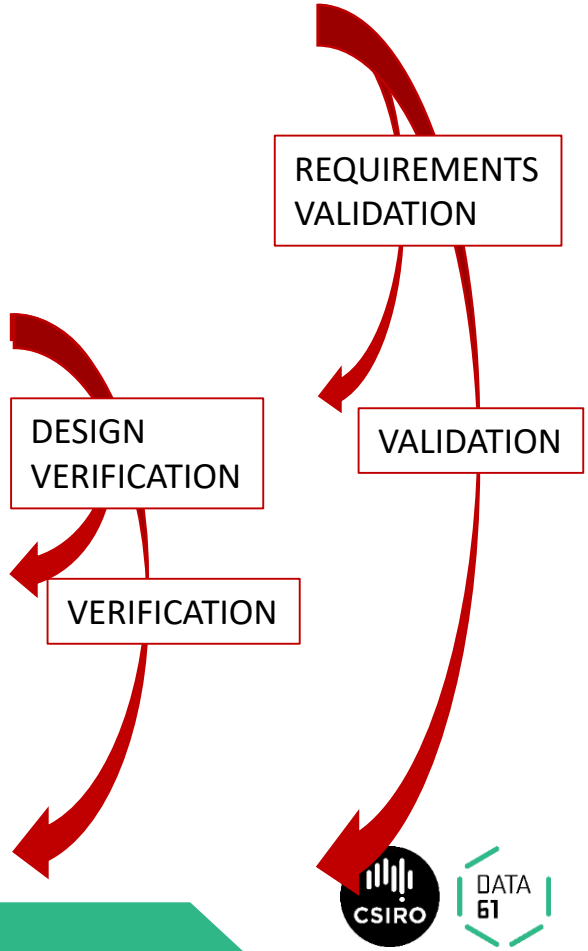
- Functional Correctness
  - degree to which a product or system provides the correct results with the needed degree of precision
- Functional Completeness
  - degree to which the set of functions covers all the specified tasks and user objectives
- Functional Appropriateness
  - degree to which the functions facilitate the accomplishment of specified tasks and objectives

Verification and Validation

证实

确认，验证





(Staples, 2014, 2015)

What Function? Is it true that “Code is Law?

- Lawrence Lessig famous quote “Code is Law”

  https://harvardmagazine.com/2000/01/code-is-law-html

  - Some people think this means “The power of code can take over from the law”
  - Lessig meant: “Code requires choices about values, has social implications, and should be governed like the law”

- “The DAO” failure on Ethereum in 2016
  - An instance of the “Distributed Autonomous (自治的) Organisation” concept
  - “The DAO” was for decentralised control of investment funds, raised > $150M
  - But the code was used in an unexpected way to siphon (虹吸管) funds away (~ $50M?)
  - Led to a hard fork in Ethereum vs. Ethereum Classic to reverse the effects of that

The DAO terms

Explanation of Terms and Disclaimer

放弃，拒绝

The terms of The DAO Creation are set forth in the smart contract code existing on the Ethereum blockchain at 0xbb9bc244d798123fde783fcc1c72d3bb8c189413. Nothing in this explanation of terms or in any other document or communication may modify or add any additional obligations or guarantees beyond those set forth in The DAO's code. Any and all explanatory terms or descriptions are merely offered for educational purposes and do not supercede or modify the express terms of The DAO's code set forth on the blockchain; to the extent you believe there to be any conflict or discrepancy between the descriptions offered here and the functionality of The DAO's code at 0xbb9bc244d798123fde783fcc1c72d3bb8c189413, The DAO's code controls and sets forth all terms of The DAO Creation.

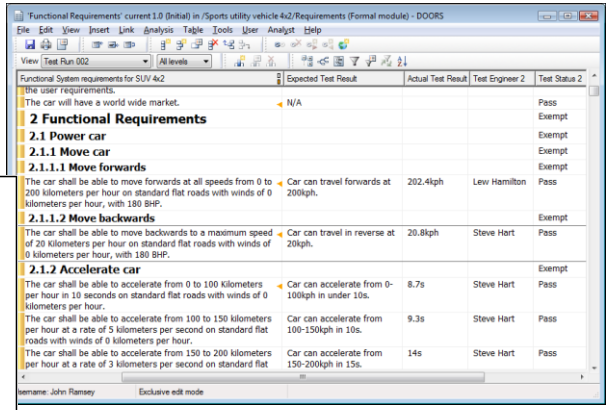
What Should Your Specification Be?

- There was no problem with Ethereum virtual machine; the code functioned exactly as it was written
- Question: Was this a “bug” in the DAO code?
- Lessons from the DAO failure
  - Lesson: Code is not the law
  - Lesson: Code is not a good way of specifying smart contracts
- Open question: how should we specify smart contracts?

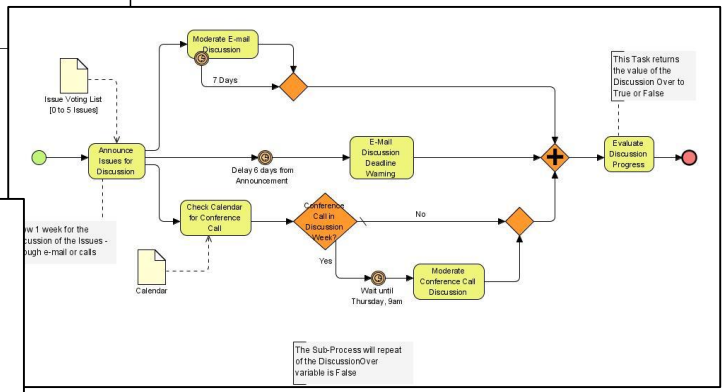
Specifications are Models of Requirements

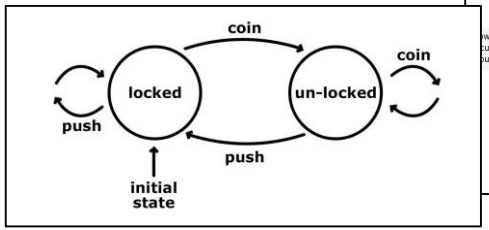
说明书

- Natural Language Specifications
  - Lists of Requirements
  - Use Cases and Scenarios
- Formal Specifications
  - Z, Object-Z, B, Event-B, HOL
- Models
  - State Machines, Business Process Models




```
MACHINE Localization
SEES TypeSets
VARIABLES
        estimated_loc
INVARIANTS
        inv1 : estimated_loc ∈ LATITUDE × LONGITUDE
EVENTS
Initialisation
        begin
                act1 : estimated_loc := null ↦ null
        end
Event LocalizeVehicle ≙
        begin
                act1 : estimated_loc :∈ (LATITUDE \ {null}) ×
                        (LONGITUDE \ {null})
        end
END
```



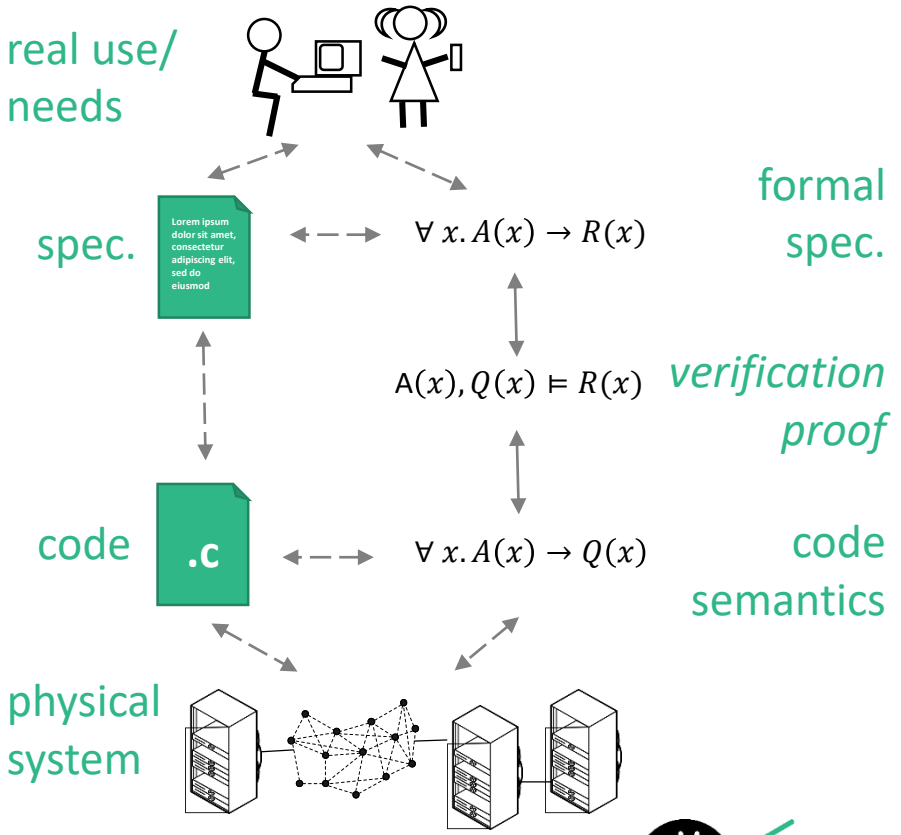


Formal Specification & Verification

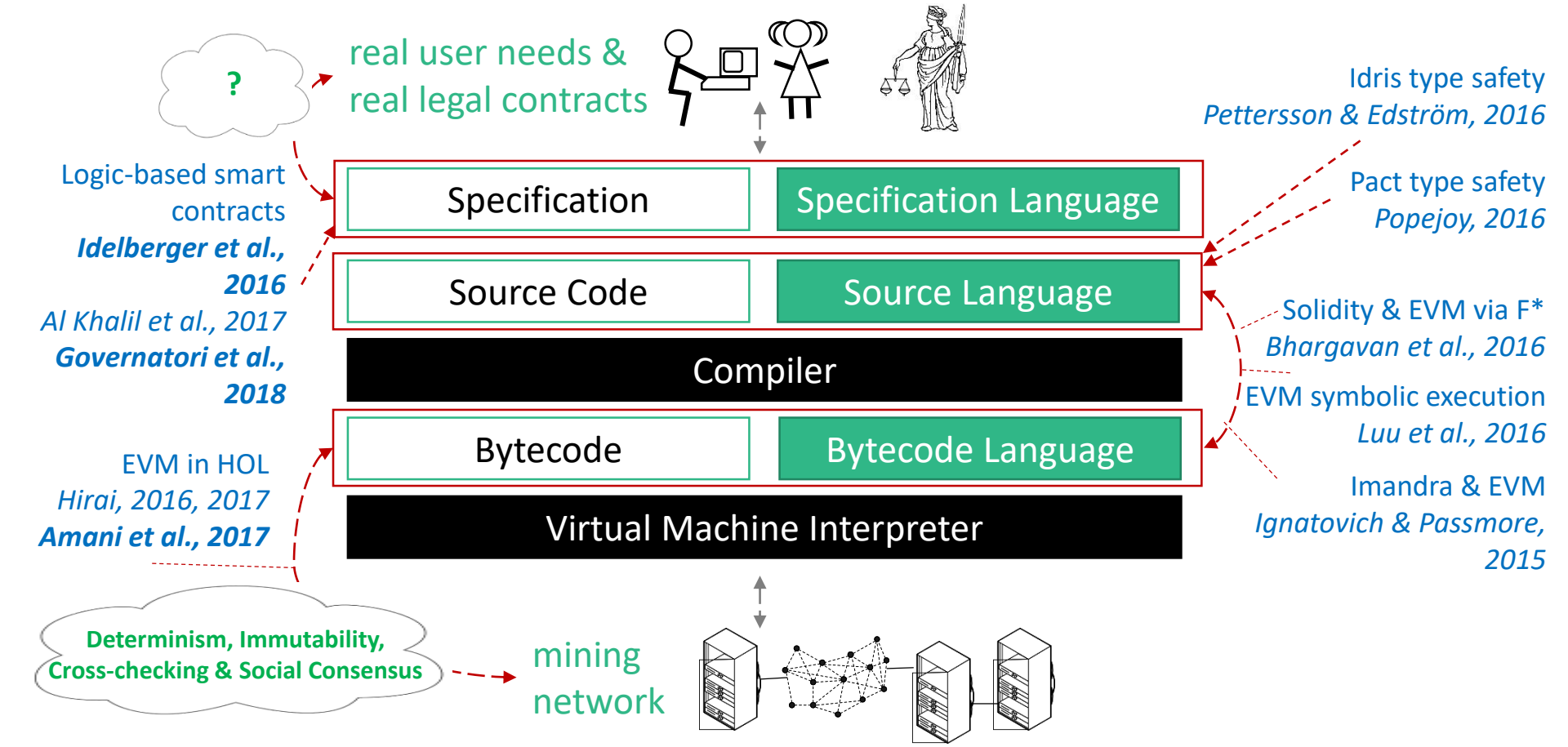
Program testing can be used to show the presence of bugs, but never to show their absence!

- Dijkstra, ’60s/’70s

- Formal methods is logical reasoning about all possible behaviours of software
- Is the strongest kind of evidence that software is correct



Formalising Smart Legal Contracts



Security

ISO/IEC 25010:2011 Security Characteristics

- Integrity
  - degree to which a system, product or component prevents unauthorized access to, or modification of, computer programs or data
- Confidentiality
  - degree to which a product or system ensures that data are accessible only to those authorized to have access
- Non-repudiation
  - degree to which actions or events can be proven to have taken place, so that the events or actions cannot be repudiated later
- Accountability 有责任，可说明性
  - degree to which the actions of an entity can be traced uniquely to the entity
- Authenticity 真实性
  - degree to which the identity of a subject or resource can be proved to be the one claimed

Only good writes & deletes

Only good reads

Undeniable

You did it!

Not fake

(Privacy is controlling access to yourself; and not just information)

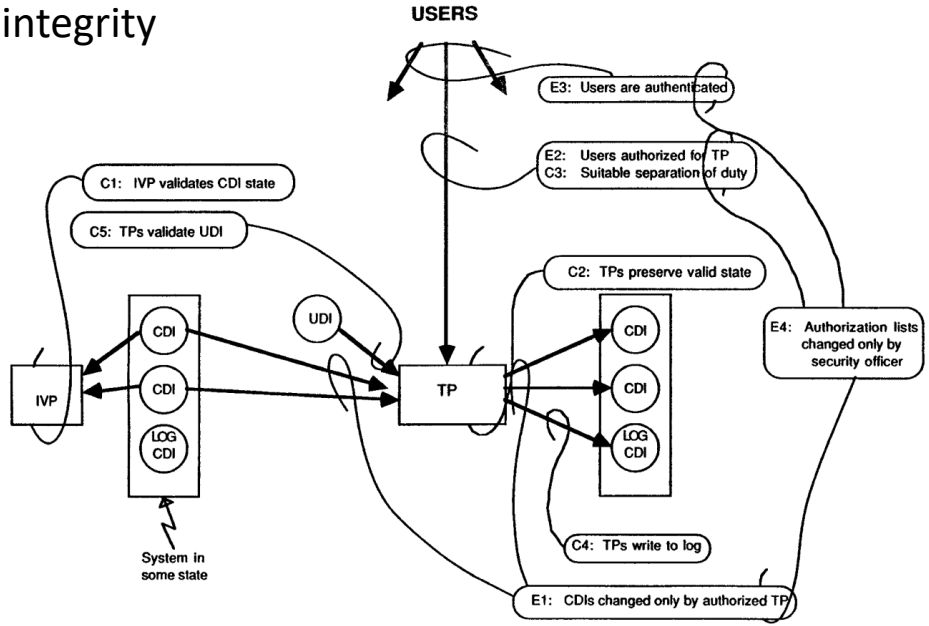
(ISO/IEC 5010:2011 treats Availability as a “Reliability” characteristic)

Integrity

- ISO/IEC 25010:2011: “degree to which a system, product or component prevents unauthorized access to, or modification of, computer programs or data”

- Clark-Wilson policy model is classic model of integrity
  - Based on double-entry bookkeeping

- Start with a well-formed initial state
- “Transformation Procedures” preserve state well-formedness
  - “Integrity Validation Procedures” check conditions for “Constrained Data Items”
- Only authenticated & authorised users can make changes
  - Also ensure appropriate “separation of duty”
- Audit logs, controls on admin changes



“A Comparison of Commercial and Military Computer Security Policies”

https://groups.csail.mit.edu/ana/Publications/PubPDFs/A%20Comparison%20of%20Commercial%20and%20Military%20Computer%20Security%20Policies.pdf

Integrity for Blockchain Platforms

- Clark-Wilson perspective on blockchain integrity
  - Blockchain ledger is the system state and the audit log
    - Blocks and their transactions are the “Constrained Data Items”
  - Initial state: genesis block is well-formed & cross-checked by all miners
  - Mining and cross-checking other miners’ blocks is the “Transformation Procedure”
  - Miners’ block validation checks are the “Integrity Validation Procedure”
  - Authorisation is checked using signatures from public-key cryptography
  - No authentication on public blockchains! Often important on private blockchains
  - No separation of duty enforced?
  - “Admin changes” are by the mining collective, e.g. hard forks

- Example integrity conditions
  - Were transactions against an account address signed by corresponding private key?
  - Does the sending address/account have enough cryptocurrency?
    - Some platforms support other crypto-assets with different integrity conditions
  - Did a miner give themselves the right mining reward?
  - Did the execution recorded for a smart contract give the same result I get?

“Blockchain anomaly”

异常

- Blockchain transactions are in a strict order, and are immutable… or are they?
- Nakamoto consensus can give us alternative/ replacement histories

1. Issue a transaction
2. Wait to you see it
3. Issue second transaction that depends on the first one
4. Possible anomaly!

Both need to be independently “valid” from the perspective of a blockchain, but invalid to the application when reordered

Solutions: wait for more confirmation blocks, try to rely n Ethereum “transaction nonce”, or use a smart contract to enforce/check the ordering

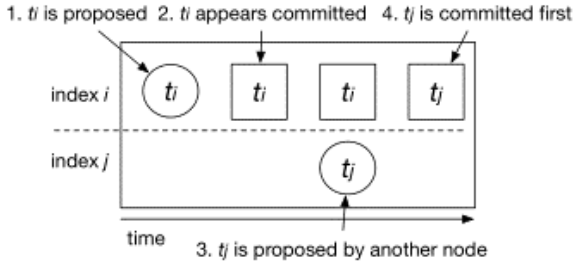


Figure 3: The Blockchain anomaly: a first client issues t_i that gets successfully mined and committed then a second client issues t_j, with t_j being conditional to the commit of t_i (note that $j \geq i + k$ for t_i to be committed before t_j gets issued), but the transaction t_j gets finally reorganised and successfully committed before t_i, hence violating the dependency between t_i and t_j

“The Blockchain Anomaly”
https://arxiv.org/pdf/1605.05438

Integrity for Component Services in Blockchain Applications

- Integrity of Data Storage and Communication
  - On-chain data in transactions: “enforce” using off-chain code & conventions
  - On-chain data in smart contract variables: enforce using smart contract code
  - Off-chain data: enforce by cross-checking data hash with on-chain hash values & also check authorisation off-chain using the signatures for those transactions
  - Off-chain data: enforce by checking data using rules stores on-chain

- Integrity of Computation and Asset Management
  - On-chain: smart contracts themselves being correct! Or to cross-check others
  - Off-chain: smart contracts cross-check off-chain computation (e.g. “state channel” pattern)

- Smart contracts can encode & enforce application-level integrity checks and transformations on-chain – code needs to be correct!
  - e.g. were the application-level preconditions true when an API call was made?
  - e.g. was the API call made by an authorised party?

Confidentiality

- ISO/IEC 25010:2011: “degree to which a product or system ensures that data are accessible only to those authorized to have access”
- Blockchain platforms are not good for confidentiality, because mining nodes cross-check contents of all transactions
- Not just the plain data, also need to be concerned with re-identification attacks, patterns from transaction analytics, and graph datamining
  - Identity of parties?
  - Transaction volume?
  - Transaction meta-data?
    - e.g. characteristic patterns in time-of-day for transactions, geolocation of source IP addresses of submitted transactions

Confidentiality for Blockchain Applications

Might uUUse…	But…
pseudonymous addresses	• if addresses are re-used, can start to recover identity information
private blockchain	• check all nodes are authorised to see all the data!
distributed ledger where ledgers and transactions are shared only with parties of interest (e.g. Corda, Hyperledger Fabric)	• can be harder to ensure property rights for digital assets (“double spending” issues)
encryption for on-chain confidential data	• more complex key management • key loss reveals all old data too • quantum? or other breaks to the crypto schemes
keep data off-chain, use conventional technology for access control	• more complex system design • less value directly from blockchain for sharing data

Privacy

- Similar but different to confidentiality

- Privacy is controlling access to yourself; and not just information
  - e.g. physical privacy in your house
  - 1948 Universal Declaration of Human Rights:
    No one shall be subjected to arbitrary interference with his privacy, family, home or correspondence, nor to attacks upon his honor and reputation. Everyone has the right to the protection of the law against such interference or attacks.

- Even for information, privacy has different regulation than confidentiality
  - General Data Protection Regulation (GDPR) in Europe
    - Including “right to erasure (right to be forgotten)”, but can’t delete on blockchain?

- For privacy in blockchain-based applications, use same general approaches as for confidentiality, targeting privacy policies

Non-Repudiation

- ISO/IEC 25010:2011: “degree to which actions or events can be proven to have taken place, so that the events or actions cannot be repudiated later”

- Main support is from blockchain’s immutable ledger
  - But be careful of probabilistic immutability in Nakamoto consensus; use the right number of confirmation blocks for your application’s risk profile
- Some support from public key signatures for transactions

- Just because data is recorded on-chain, doesn’t mean it’s true!
  - Someone might have stolen my private key?
  - Sender might have fraudulently recorded false data in a transaction
- Only have non-repudiation that the transaction happened

- If using hashes on-chain for off-chain data, need to retain original data

Accountability

- ISO/IEC 25010:2011: “degree to which the actions of an entity can be traced uniquely to the entity”
- Main support in blockchain platforms is from cryptographically-signed transactions
  - Does a single entity control the signing key? Good key management is critical!
- Most public blockchains try to directly stop this
  - Pseudonymous IDs are too weak to
  - New privacy coins (Zcash, Dash, Monero, …) try to provide anonymity like cash
- KYC/AML-CTF
  - “Know Your Customer”/ “Anti-Money Laundering, Counter-Terrorism Financing”
  - e.g. Australian AUSTRAC regulation requires KYC/AML by crypto-currency exchanges
- Private blockchains are normally permissioned, and know who users are
- Authentication requires off-chain mechanisms to establish identity
- Blockchain can be used to communicate or ensure integrity of KYC information

Authenticity

- ISO/IEC 25010:2011: “degree to which the identity of a subject or resource can be proved to be the one claimed”

- Major issue in supply chain, e.g. is this real Australian baby formula?
- Many blockchain applications are trying to improve supply chain quality

- For physical assets, hard to guarantee, unless a unique physical “fingerprint”
  - e.g. diamonds, DNA?
- Attacks include fake duplicate labels, substitution in (or reuse of) original packaging
- Need a way of comparing blockchain record to the physical item

- For digital assets, much easier on blockchain!
  - Put authoritative register on blockchain, or have an off-chain authority sign the a[illegible]

Reliability

CSIRO DATA 61

ISO/IEC 25010:2011 Reliability Characteristics

- Reliability
  - degree to which a system, product or component performs specified functions under specified conditions for a specified period of time
- Availability
  - degree to which a system, product or component is operational and accessible when required for use
- Recoverability
  - degree to which, in the event of an interruption or a failure, a product or system can recover the data directly affected and re-establish the desired state of the system
- Maturity
  - degree to which a system, product or component meets needs for reliability under normal operation
- Fault-Tolerance
  - degree to which a system, product or component operates as intended despite the presence of hardware or software faults

Availability

- ISO/IEC 25010:2011: “degree to which a system, product or component is operational and accessible when required for use”
  - A measure could be something like “probability of being able to provide service at any given time” or “
  - Affected by the other reliability characteristics, such as probability of failure, fault tolerance, time to recovery, etc

- Blockchain platform is highly redundant (many nodes)
- Easy to subscribe to updates to get replicas of the ledger
- An application can run many redundant blockchain nodes locally
  → high read availability

- Write availability is a different story…

Latency Variability Can Limit Write Availability

- Scenario
  - your application sends a transaction to update blockchain ledger
  - waiting… waiting…
  - ? should I keep waiting? or was there a service failure? (i.e. an availability problem)

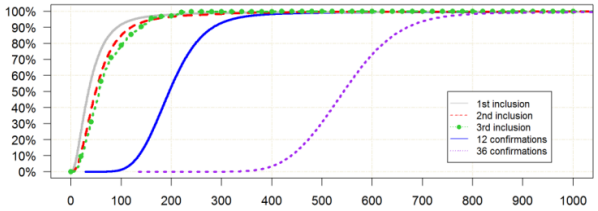


Figure 5: Time (sec) for first inclusion and commit (12 or 36 confirmations), as well as second and third inclusion of transactions that were previously not included in main chain.

- You will define some limit on your waiting time
- Long tail of variability in latency means sometimes it will take even longer
- But, by then your application will have decided a failure has happened

- Complex trade-off:
  - Availability vs. Cost (gas/fee) vs. Latency (# confirmations) vs. Risk (# confirmations)

- Mainly a problem with public blockchains?
  - (Depending on consensus mechanism, mining node policies, etc)

"On availability for blockchain-based systems"
https://ieeexplore.ieee.org/document/8069069

Availability for Blockchain-Based Applications

- A blockchain-based application has many components
  - Even if the blockchain platform works, your other components may fail

- Use normal availability-increasing design strategies for the architecture, for example…
  - Increase quality of each component & connector
    - High quality software and hardware (!)
  - Eliminate single points of failure/ increase redundancy
    - Load balancing/failover monitoring and routing
    - Stateless server components
      - Blockchain can help enable “stateless” server component (use blockchain to store the state)
  - Detect and recover from failures
    - Hot backup/failover servers

Recoverability

- ISO/IEC 25010:2011: “degree to which, in the event of an interruption or a failure, a product or system can recover the data directly affected and re-establish the desired state of the system”

- Blockchain platform is likely to recover – consensus mechanisms are designed to autonomically recover nodes and ledger consistency
  - In worst case in a public blockchain, there might be a hard fork in the ledger
  - Normally, recoverability will mostly be a concern for other parts of the application architecture

- But, can you abort a blockchain transaction, to safely retry?
  - Most blockchains do not explicitly provide a support for this!

How to Abort a Transaction in Ethereum?

中止

- If you send a transaction into the pool, and it's not yet included, can you abort it?
  - You've changed your mind, or something is wrong, or taking too long to commit
- Ethereum transactions have a nonce, normally increment per transaction
  - Miners will consider an old or reused nonce as being "outdated"

1. Issue a competing null transaction with same address & transaction nonce, with a higher fee
   - e.g. send 0 Ether to yourself, or invoke a smart contract to raise an exception
2. Reissue the "same" transaction from same address & nonce, with a higher fee
   - Higher fee means it has different hash value, so will be seen as "different"

Does Transaction Aborting in Ethereum Work?

- It’s not guaranteed to work
- This trick creates a race condition; either transaction in the pool might get included
  - Depends on not just the fee for the new transaction, but also the fee for the initial transaction
- Also unclear if transaction nonce is always honoured by miners (c.f. “Blockchain anomaly” discussed earlier)

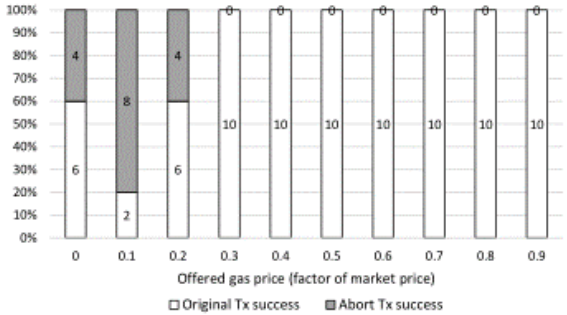


Figure 11: Underbidding market fee and automatic abort after 10 minutes if the original Tx was not included

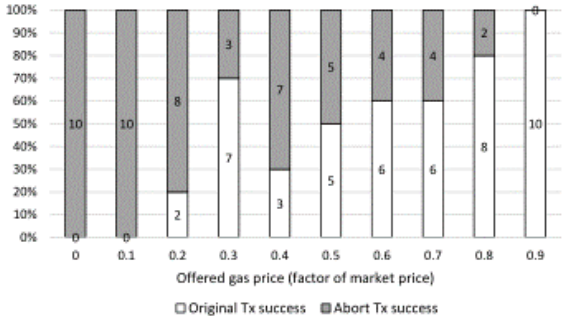


Figure 12: Underbidding market fee and automatic abort after 3 minutes if the original Tx was not included

“On availability for blockchain-based systems”
https://ieeexplore.ieee.org/document/8069069

Maturity (“Reliability”)

成熟

- ISO/IEC 25010:2011: “degree to which a system, product or component meets needs for reliability under normal operation”
  - *My opinion: not a great name for this…*

- Availability is about readiness for correct service

vs.

- Reliability is about continuity of correct service

- Previous discussion about availability for blockchain-based applications applies here too

Fault-Tolerance

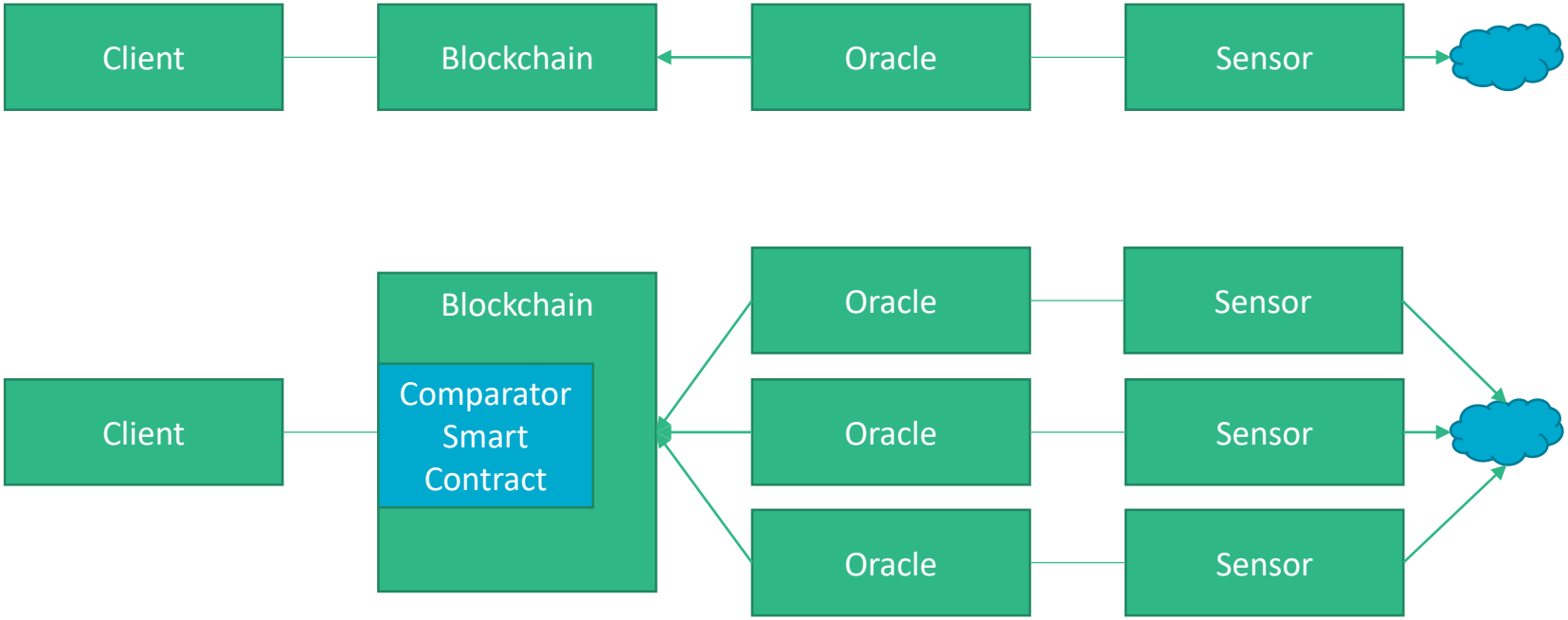
- ISO/IEC 25010:2011: “degree to which a system, product or component operates as intended despite the presence of hardware or software faults”

- Blockchain platform tolerates faults
  - Hardware faults in individual miners: other miners will keep working
  - Software faults too, if there is enough diversity of mining software
    - In worst case, might get a hard fork

- For blockchain applications
  - Use normal design strategies for fault tolerance (redundancy, monitor, detect, isolate recover) for other parts of the architecture
  - Smart contracts are not normally fault-tolerant for software faults
  - Smart contracts can be used to do fault tolerance logic for off-chain components…

Oracle Faults

- Can use on-chain M-of-N voting for redundant oracles



Summary

Blockchains as Dependable Systems

- Trust – accepted dependence
- Trustworthy – justified Trust
- Need Evidence to justify Engineering Assurances

- Validation
  - Sort-of-but-not-quite: “am I solving the right problem?”
  - Does my requirements/specification/system/design address/solve the user problem?
- Verification
  - Sort-of-but-not-quite: “am I solving the problem right?”
  - Does my system/design meet the requirements specification?

- Can provide evidence based on architectural analysis, early in the lifecycle
- Blockchains are increasingly relied on; need evidence they are trustworthy

ISO/IEC 25010:2011 Dependability & Security

- Functional Suitability
  - Correctness
  - Completeness
  - Appropriateness
    - Code is Not Law
    - How to specify smart contracts?
- Security
  - Integrity & Clark-Wilson policy model
    - Blockchain “anomaly” in Nakamoto consensus: possible transaction reordering
  - Confidentiality
    - And its difference to Privacy
  - Non-Repudiation
  - Accountability
    - Avoid or support? KYC/AML-CTF
  - Authenticity
- Reliability
  - Availability: readiness for service
    - Affected by latency variability
  - Recoverability
    - Aborting transactions
  - “Maturity”
    - Reliability: continuous service
  - Fault Tolerance
    - Use of smart contracts for redundant oracles

Assignment 2

Design & Evaluation of a Blockchain-Based System

- THIS IS NOT A PROGRAMMING ASSIGNMENT: DO NOT WRITE CODE
- WE GIVE YOU OUTLINE OF A SYSTEM AND 4 NFPS/REQUIREMENTS
  - If you need more assumptions or context, then make and describe them
  - Totally OK to research more info about the problem domain – provide citations/references
- SHOW US 2 DESIGNS
  - One using private blockchain, one using public blockchain
- HIGH-LEVEL EVALUATION OF THE TWO DESIGNS
  - Using ATAM “Scenarios” & “Utility Tree” structures, but in a spreadsheet not a picture of a tree

电子表

逻辑依据

- SUMMARY CHOICE & RATIONALE BETWEEN 3 DESIGNS
  - Conventional technology, private blockchain, public blockchain

THANK YOU

www.data61.csiro.au