



# Blockchain in Software Architecture 2: NFPs and design trade-offs

Mark Staples

Email: [mark.staples@data61.csiro.au](mailto:mark.staples@data61.csiro.au)

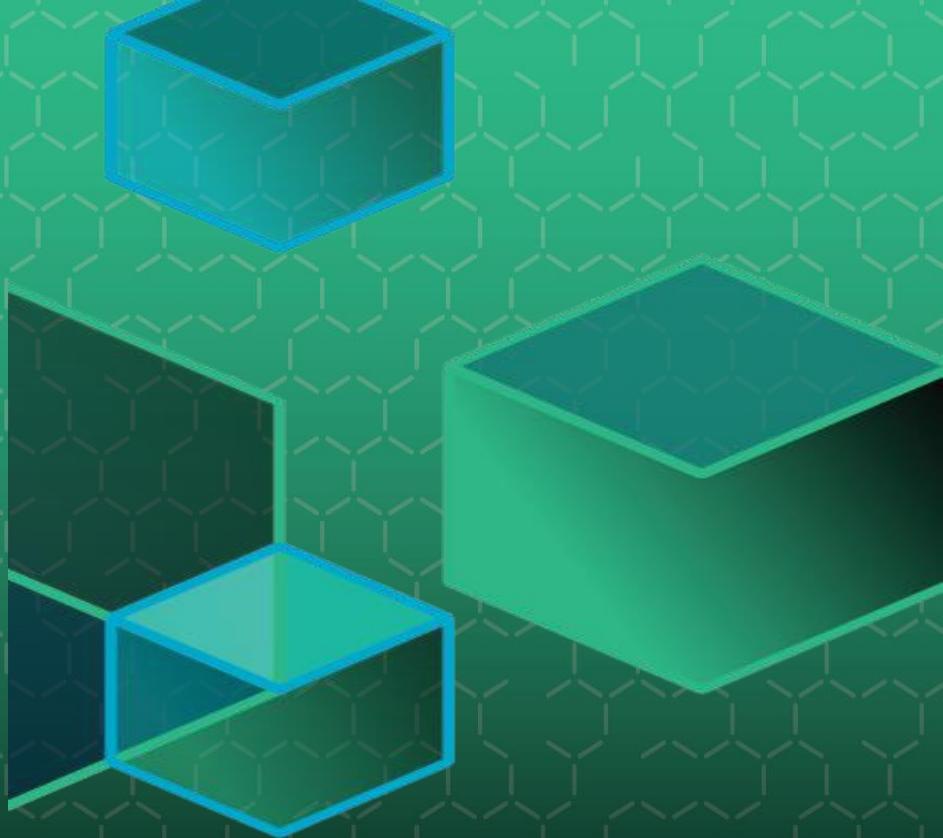
[www.data61.csiro.au](http://www.data61.csiro.au)

# Outline

- PITCH SESSION!
- What is Software Architecture?
  - Components, Connectors, Configuration
  - Non-Functional Properties (NFPs)
  - Models: Views and Viewpoints
  - NFP Analysis and Trade-offs
- Blockchain in Software Architecture
  - Blockchain as Component, Connector, Configuration
  - Blockchain Component Services
- Design and Trade-offs in Blockchain-based Applications
- Summary



# What is Software Architecture?



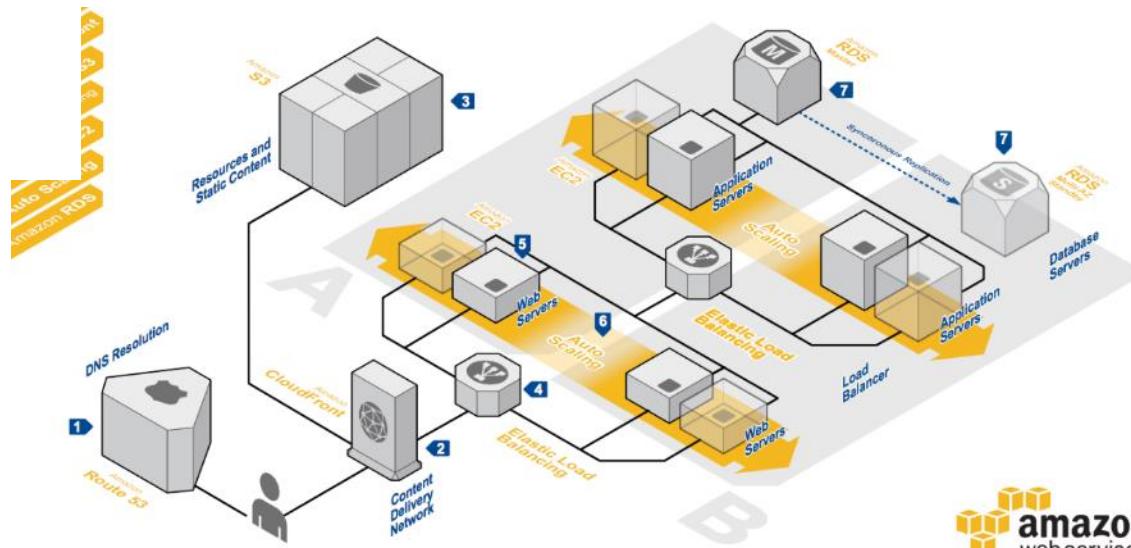
# Every software system has a software architecture

- “*The set of structures needed to reason about the system, which comprises software elements, relations among them, and properties of both*”
- Diagrams/ Documentation
- Designing
  - (But not all design is architecture)
- Discipline/ Practice/ Profession
- Discovery (Research Area)
  - Starting in 1960s, increasing interest since 1990s



# Software Architecture Elements

- A software system's architecture is typically not a uniform monolith
  - Component, Connector, and Configuration



# Software Component

- Software components are the fundamental building blocks for software architecture
- A software component is an architectural entity that
  - Encapsulates a subset of the system's functionality and/or data
  - Restricts access to that subset via an explicitly defined interface
  - Has explicitly defined dependencies on its required execution context
- Components typically provide application-specific services

# Software Connectors

- In complex systems interaction may become more important and challenging than the functionality of the individual components
- Architectural building block tasked with effecting and regulating interactions among components
- Connectors typically provide application-independent interaction facilities
  - **Communication:** transfer data
  - **Coordination:** transfer control
  - **Facilitation:** enable and optimise component's interactions
  - **Conversion:** adjust the interactions between incompatible interfaces

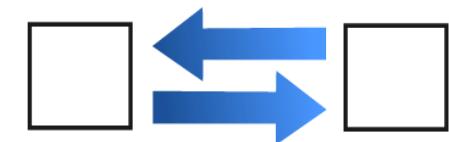
# Example Software Connectors



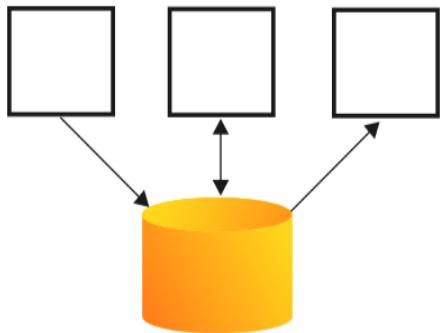
File Transfer



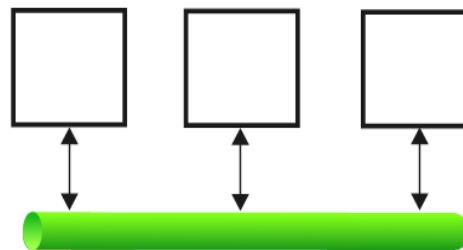
Stream



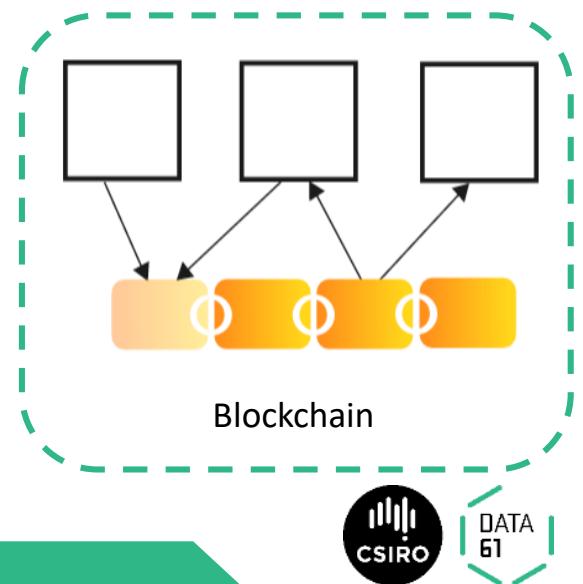
Remote Procedure Call



Shared Database



Message Bus



Blockchain

# Architecture defines system structure

- Decomposition of system into components/modules/subsystems
- Architecture defines:
  - Component interfaces
    - What a component can do
  - Component responsibilities
    - Precisely what a component will do when you ask it
  - Component connections and dependencies
    - How components are related to each other through their interfaces
- Can we infer system properties from element properties?

# Architecture specifies component communication

- Data passing mechanisms, e.g.:
  - Function call
  - Remote method invocation
  - Asynchronous message

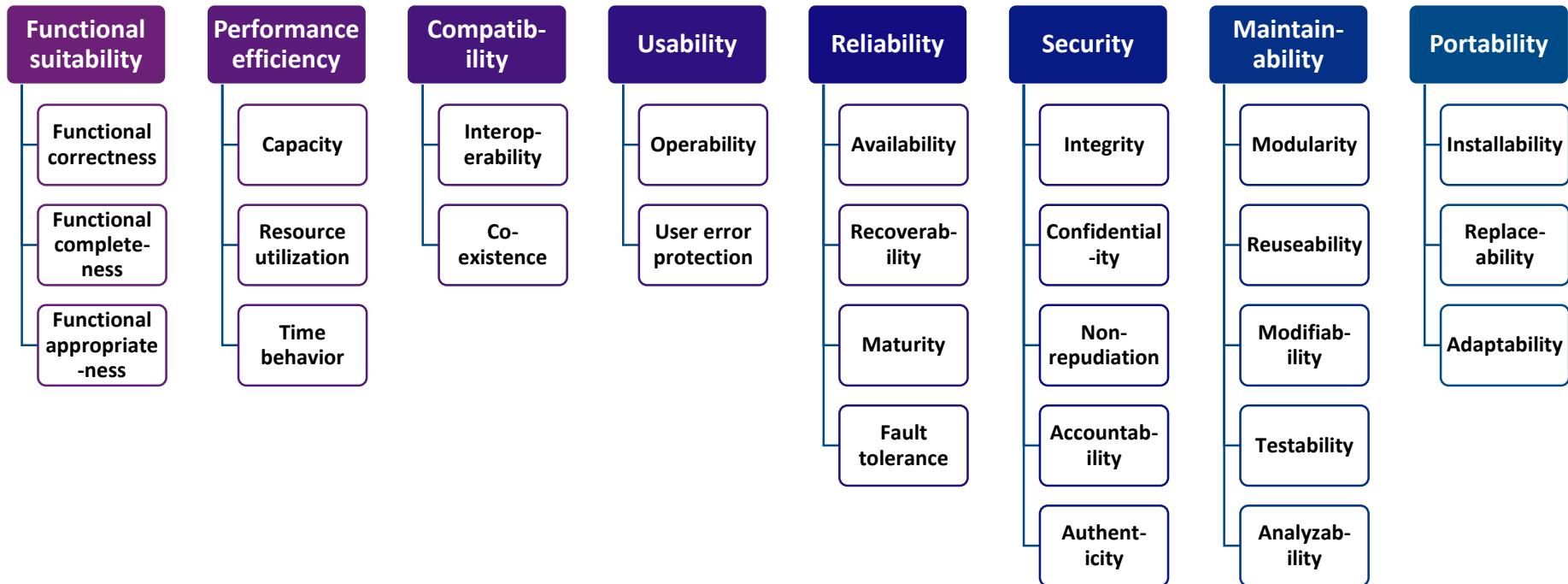
1.通过call function  
2.通过 远程方法调取  
3. 异步消息
- Control flow
  - Flow of messages between components
  - Sequential
  - Concurrent/parallel
  - Synchronization

控制流  
1.消息流  
2.序号  
3.并行  
4.同步

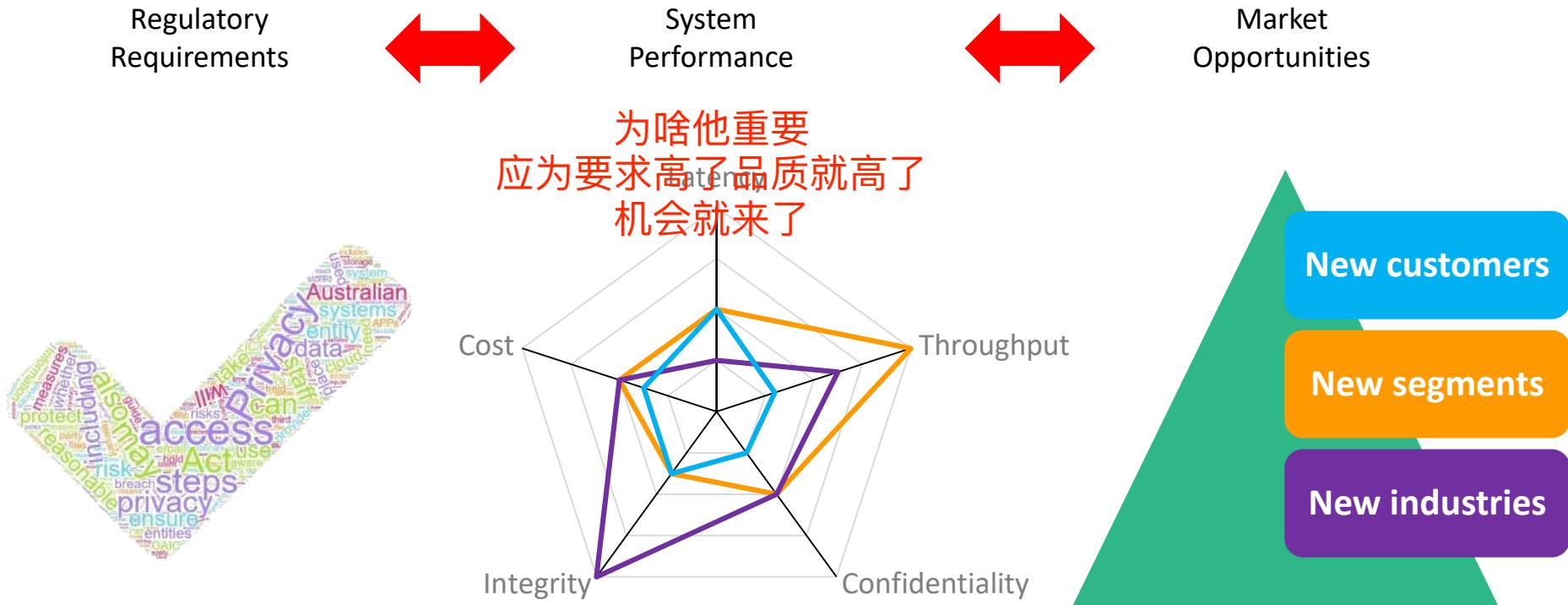
# Non-Functional Properties and Requirements

- There are two kinds of requirements:
    1. Functional Requirements (i.e. what are the inputs and outputs) **foucs on 输入和输出**
    2. Non-Functional Requirements (a.k.a. *Qualities*, or *-ilities*)
      - e.g. “Performance” (latency, throughput, ... )
      - e.g. “Security” (confidentiality, integrity, availability, privacy, ...)
      - e.g. Usability, Reliability, Modifiability, ...
      - Cost
      - Technical and business constraints
- 表示这个有什么特性,such as 安全性,可用性,可修改性

# ISO/IEC 25010:2011 Quality Model



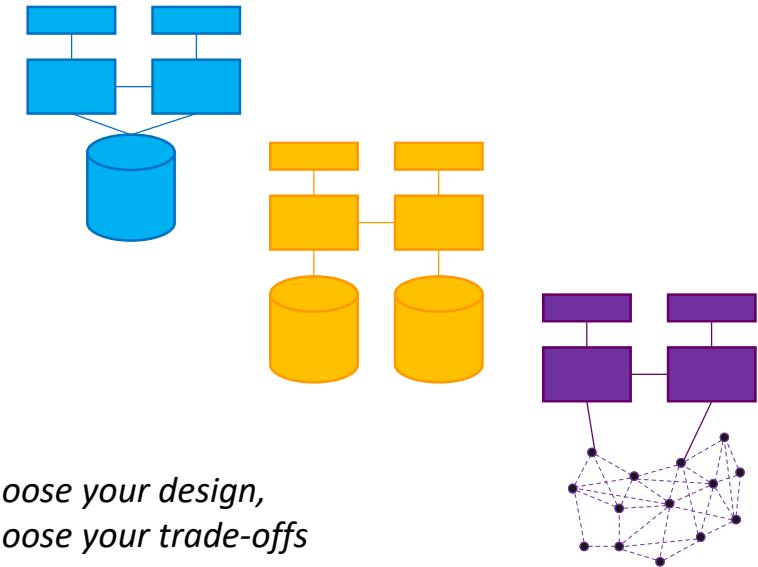
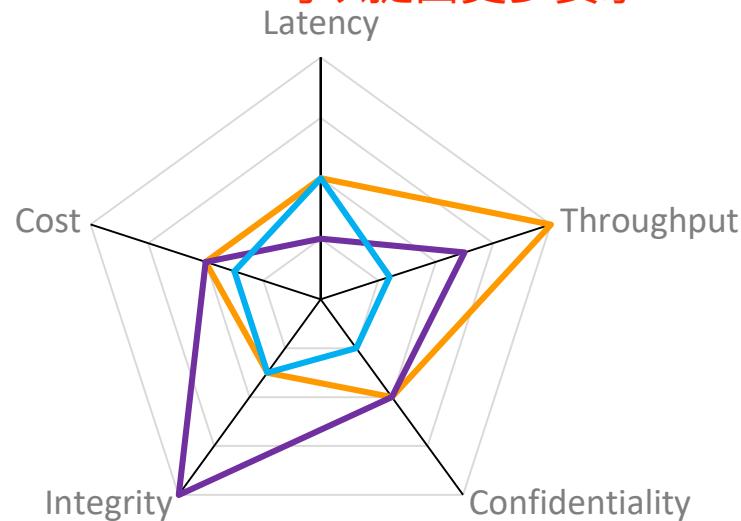
# Why Non-Functional Properties Matter



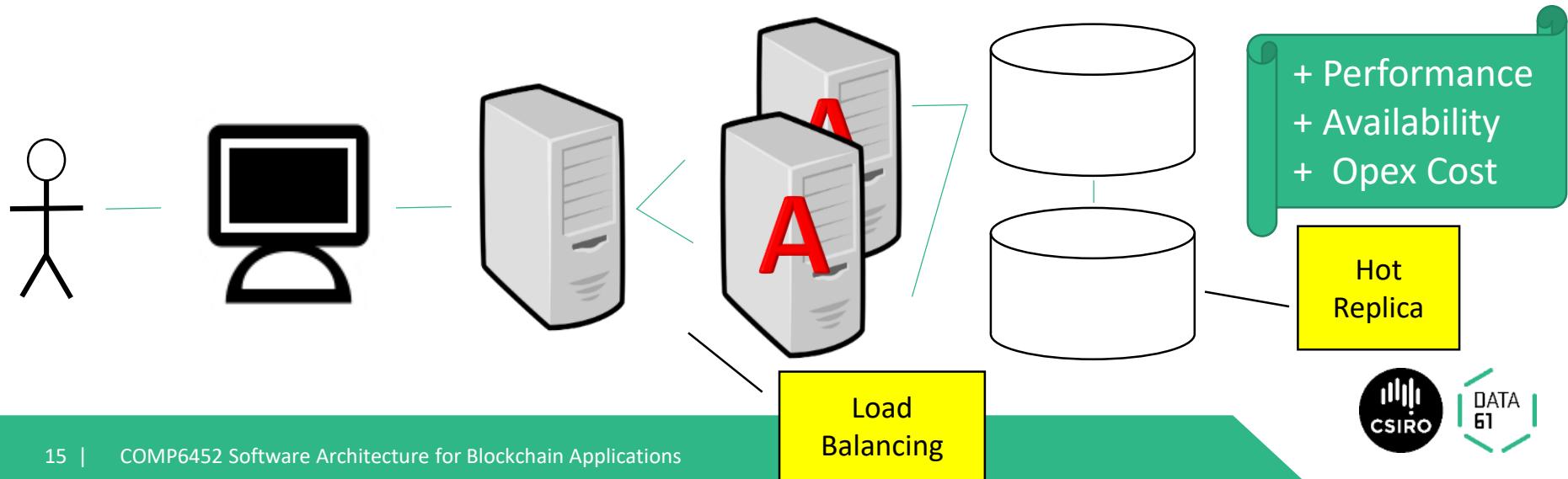
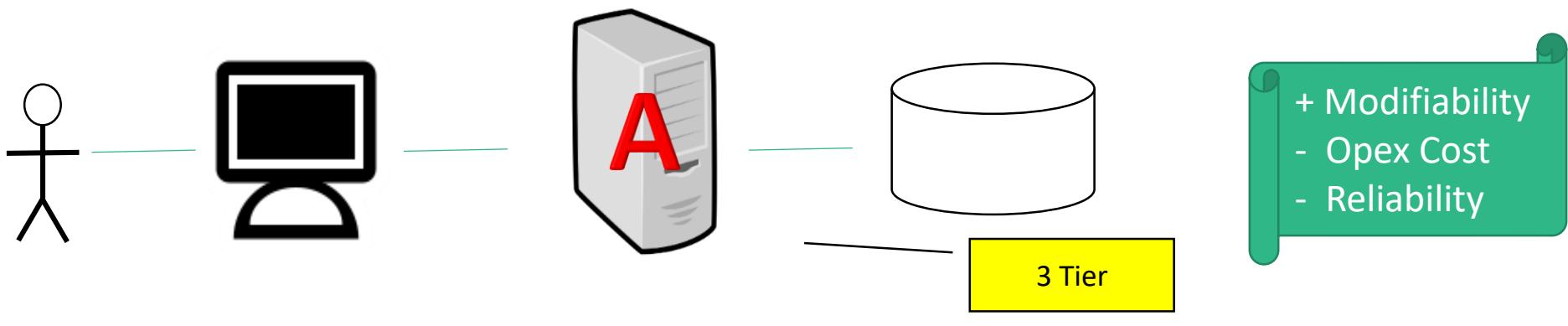
# Software Architecture

Non-Functional Properties arise from Architectural Design Choices

根据non-functional可以提出更多要求



# For Example...



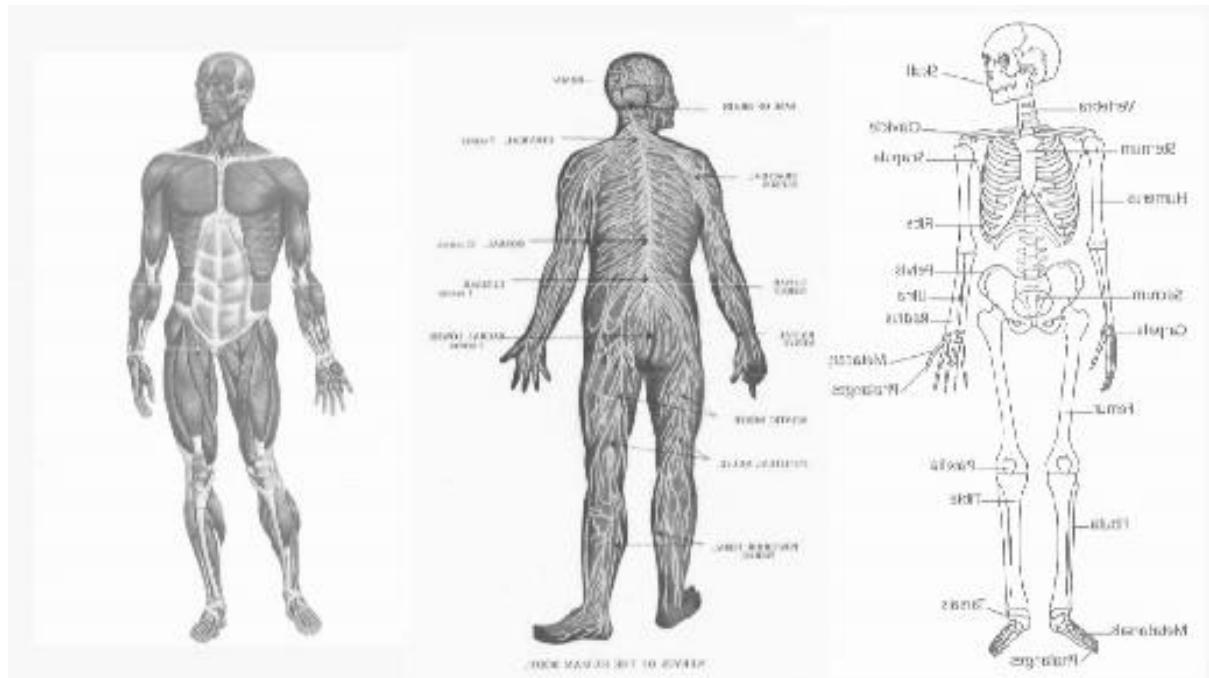
# Architecture is an Abstraction

架构是抽象化的

- Architecture provides an abstract view of a design
  - Hides complexity of design
  - May or may not be a direct mapping between architecture elements and software elements
    - e.g. “marketecture”
- “All models are wrong, but some models are useful”
  - Box
- Discussion: Why Abstraction?

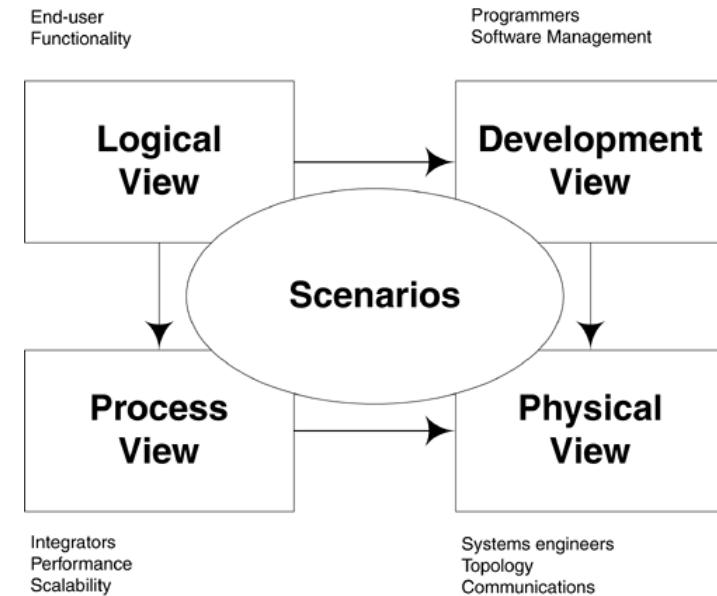


# Viewpoints and Views

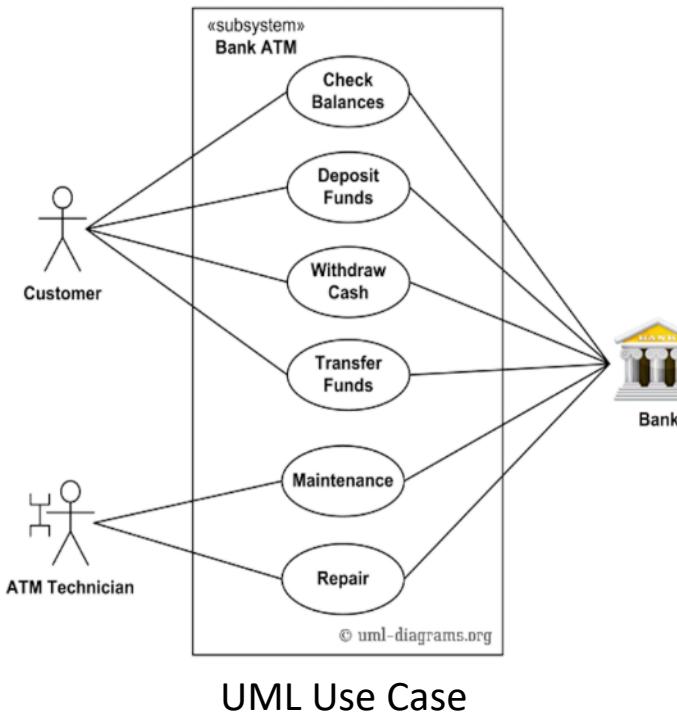


# Krutchens 4+1 View Model

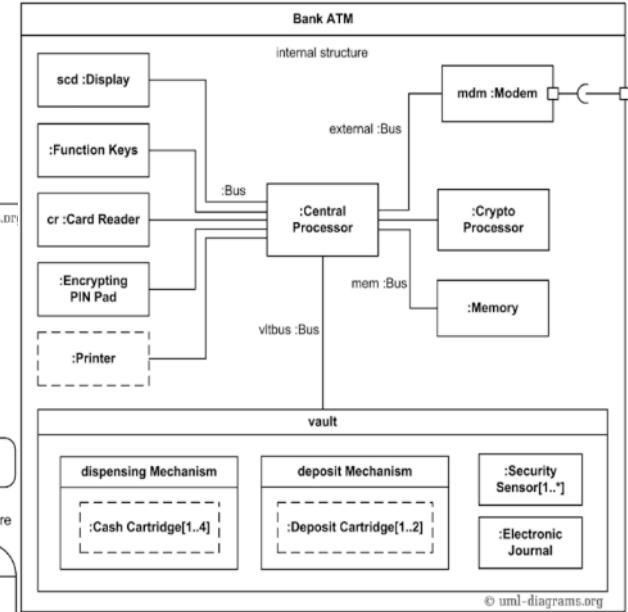
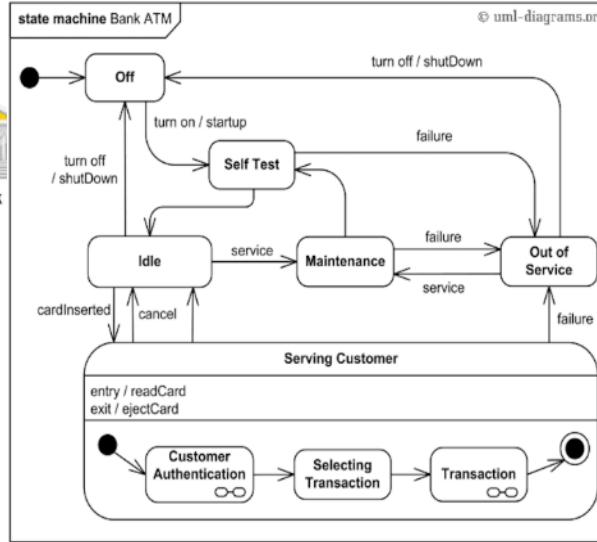
- **Logical:** architecturally-significant elements and the relationships between them.
- **Process:** concurrency and communications elements
- **Physical:** how the major processes and components are mapped to applications hardware
- **Development:** internal organization of the software components as held in e.g. a configuration management tool
- **Use cases:** requirements for the architecture; related to more than one particular view



# UML as Viewpoints and Views



UML Statechart



UML Composite Structure

# Architecture Analysis Methods

- Analysis is one of the important uses of models
- e.g. does this design support service availability?
  - (Quantitative) fault-tree statistical analysis
  - (Qualitative) failure scenarios
- e.g. What is the transaction latency in this design?
  - (Quantitative) Simulation-based prediction of latency distributions
  - (Quantitative) Formula-based calculation of average latency
- e.g. Will this design ensure confidentiality?
- e.g. Is this design easily modifiable?
- Different kinds of models allow different kinds of analyses

# Design Trade-offs

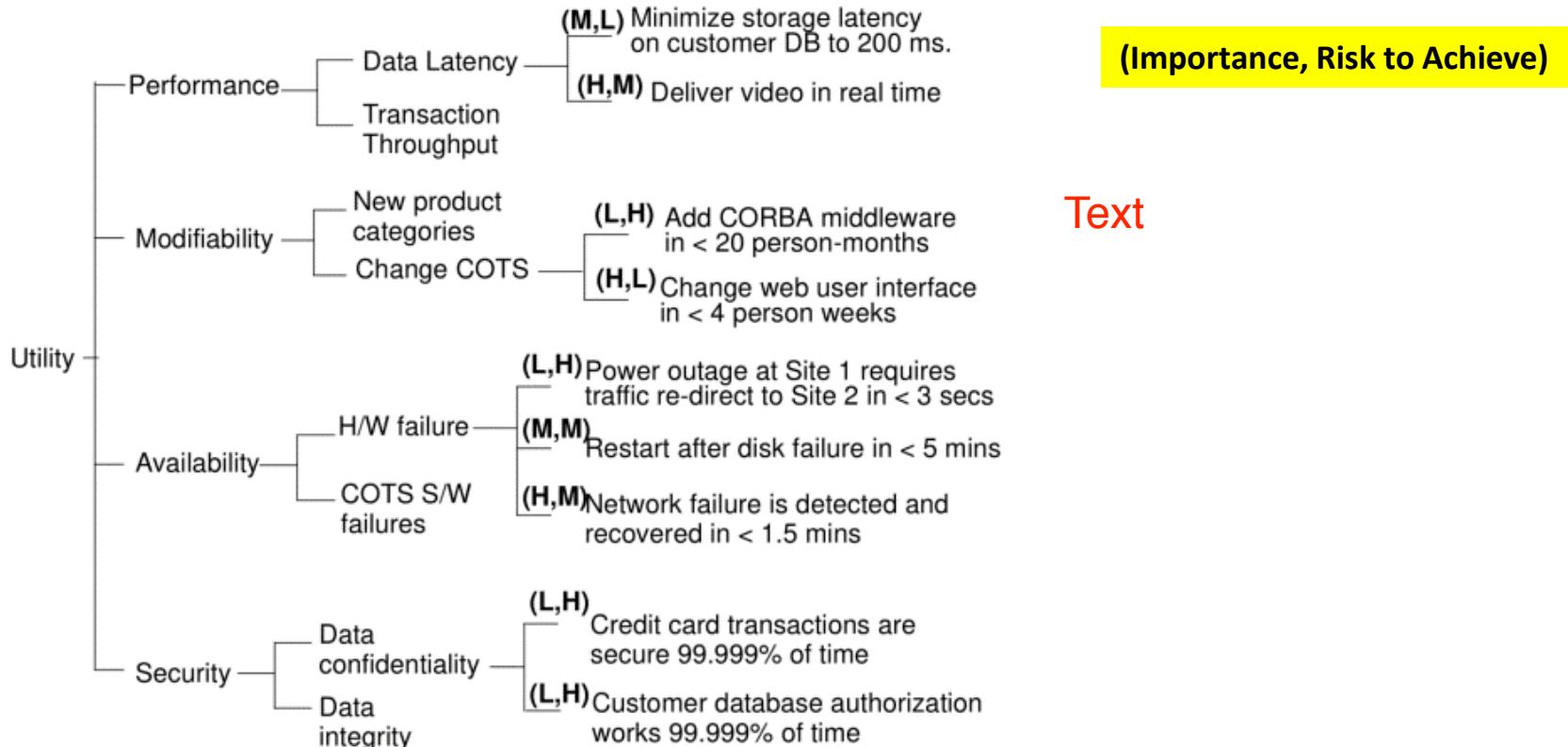
- Often, improving one quality will hurt another one
  - e.g. Use a bigger server
    - faster, but more expensive
  - e.g. Use redundant servers
    - better availability (hot fail-over)? Or better performance (load balancing)? Worse write latency? more expensive
- What are the trade-offs? How should you choose?
- Specific methods exist to help
  - ATAM: Architecture Trade-off Analysis Method
  - Multi-criteria decision-making methods

# ATAM

- Presentation
  1. Present ATAM
  2. Present Business Drivers (NFPs and other business goals)
  3. Present Architecture
- Investigation & Analysis
  4. Identify Architectural Approaches
  5. Generate Quality Attribute Utility Tree (& initial use case scenarios)
  6. Analyse Architectural Approaches
  7. Brainstorm & Prioritise Scenarios
  8. Analyse Architectural Approaches (using scenarios as test cases)
- Reporting
  9. Present Results (summary, risks, sensitivities/trade-offs, ...)



# ATAM Quality Attribute Utility Tree



Example from “ATAM: Method for Architecture Evaluation” (Kazman et al., 2000)

[https://resources.sei.cmu.edu/asset\\_files/TechnicalReport/2000\\_005\\_001\\_13706.pdf](https://resources.sei.cmu.edu/asset_files/TechnicalReport/2000_005_001_13706.pdf)

# ATAM Scenarios

**Scenario:** S12 (Detect and recover from HW failure of main switch.)

**Attribute:** Availability

**Environment:** normal operations

**Stimulus:** CPU failure

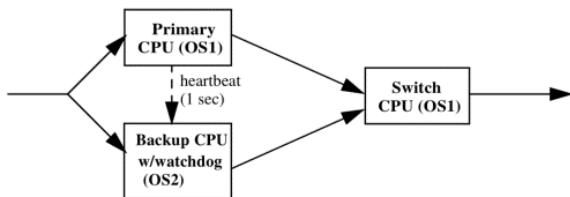
**Response:** 0.999999 availability of switch

Architectural decisions	Risk	Sensitivity	Tradeoff
Backup CPU(s)	R8	S2	
No backup Data Channel	R9	S3	T3
Watchdog		S4	
Heartbeat		S5	
Failover routing		S6	

**Reasoning:**

- ensures no common mode failure by using different hardware and operating system (see Risk 8)
- worst-case rollover is accomplished in 4 seconds as computing state takes ...
- guaranteed to detect failure with 2 seconds based on rates of heartbeat and watchdog ...
- watchdog is simple and proven reliable
- availability requirement might be at risk due to lack of backup data channel ... (see Risk 9)

Architecture diagram:



- Scenarios are commonly used in Software Architecture

- In ATAM, links attributes, risks, trade-offs, & reasoning about architecture design

- Qualitative – identify and evaluate risks

- Methodical, but not exhaustive

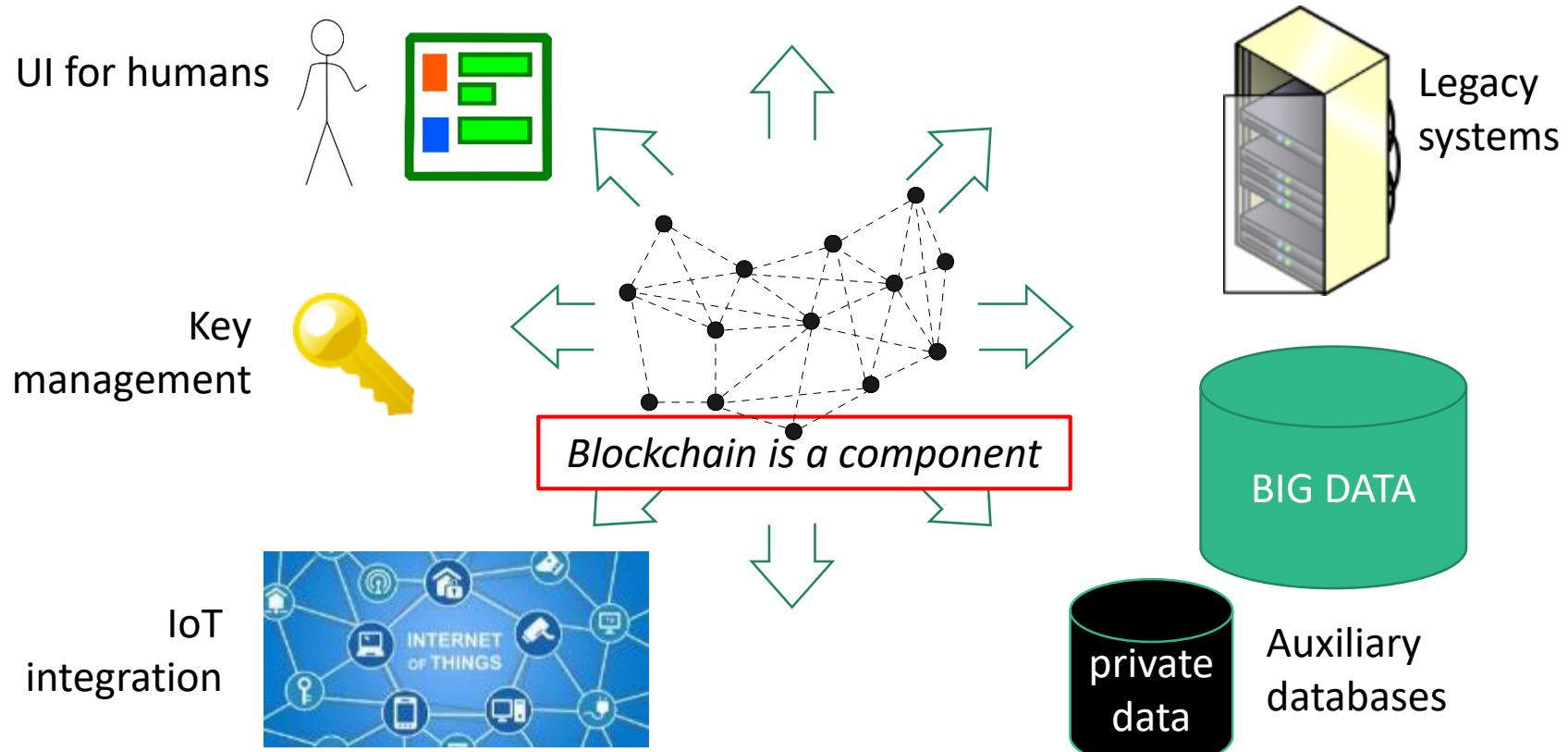
Example from “ATAM: Method for Architecture Evaluation” (Kazman et al., 2000)

[https://resources.sei.cmu.edu/asset\\_files/TechnicalReport/2000\\_005\\_001\\_13706.pdf](https://resources.sei.cmu.edu/asset_files/TechnicalReport/2000_005_001_13706.pdf)

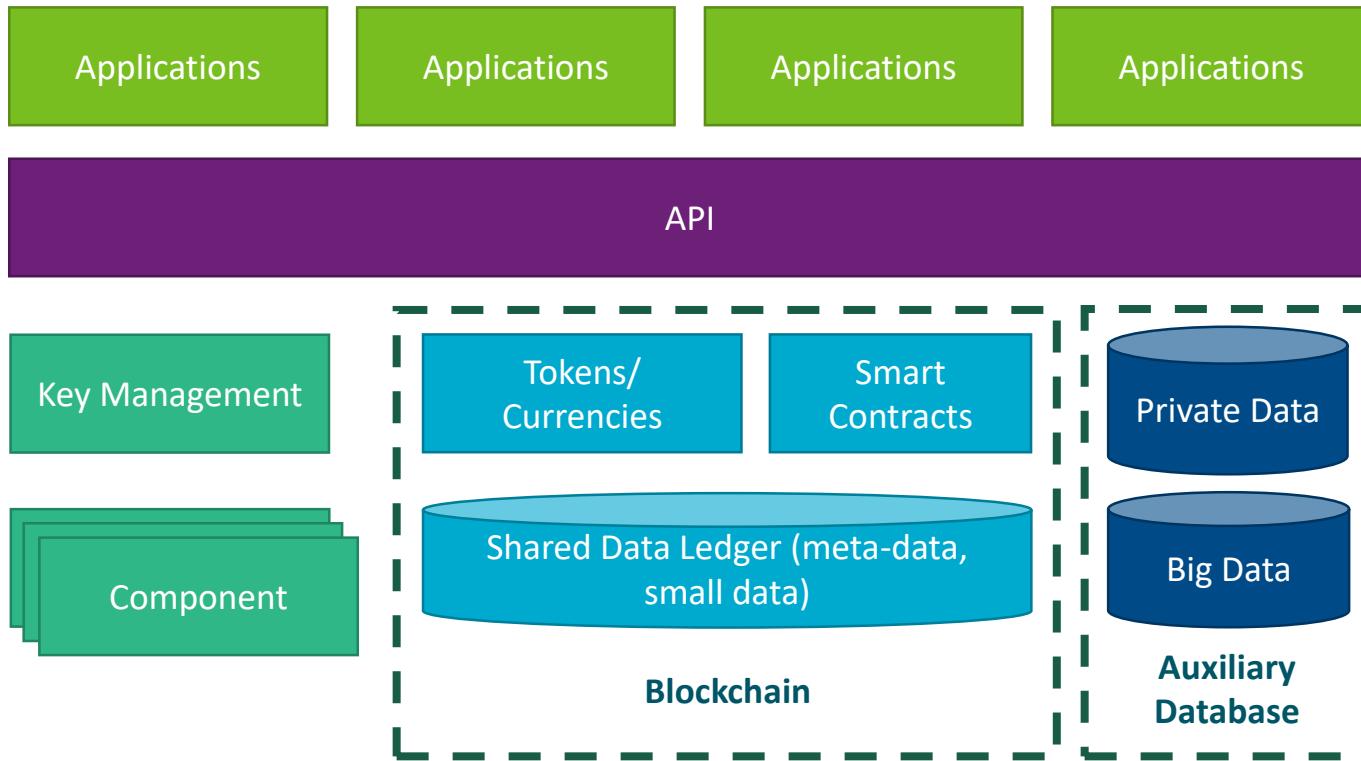
# Blockchain in Software Architecture



# Blockchains are Not Stand-Alone Systems



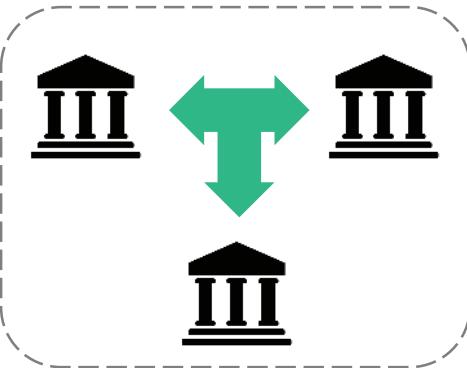
# Blockchain in Larger Software System



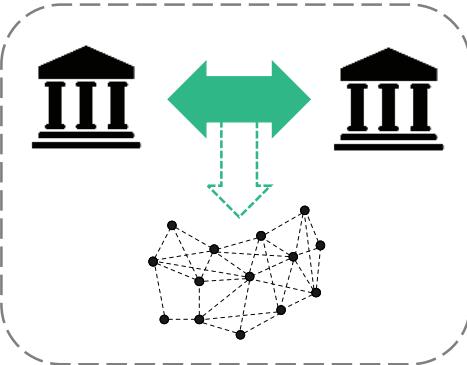
# What Does a Blockchain Do?

- Functionally, blockchains are...
- A database (ledger)
  - Record of transactions
- A compute platform
  - “Smart contracts”
- Distributed, and no central owner

Centralised Trust  
using a  
Third-Party



Distributed Trust  
using a  
Blockchain

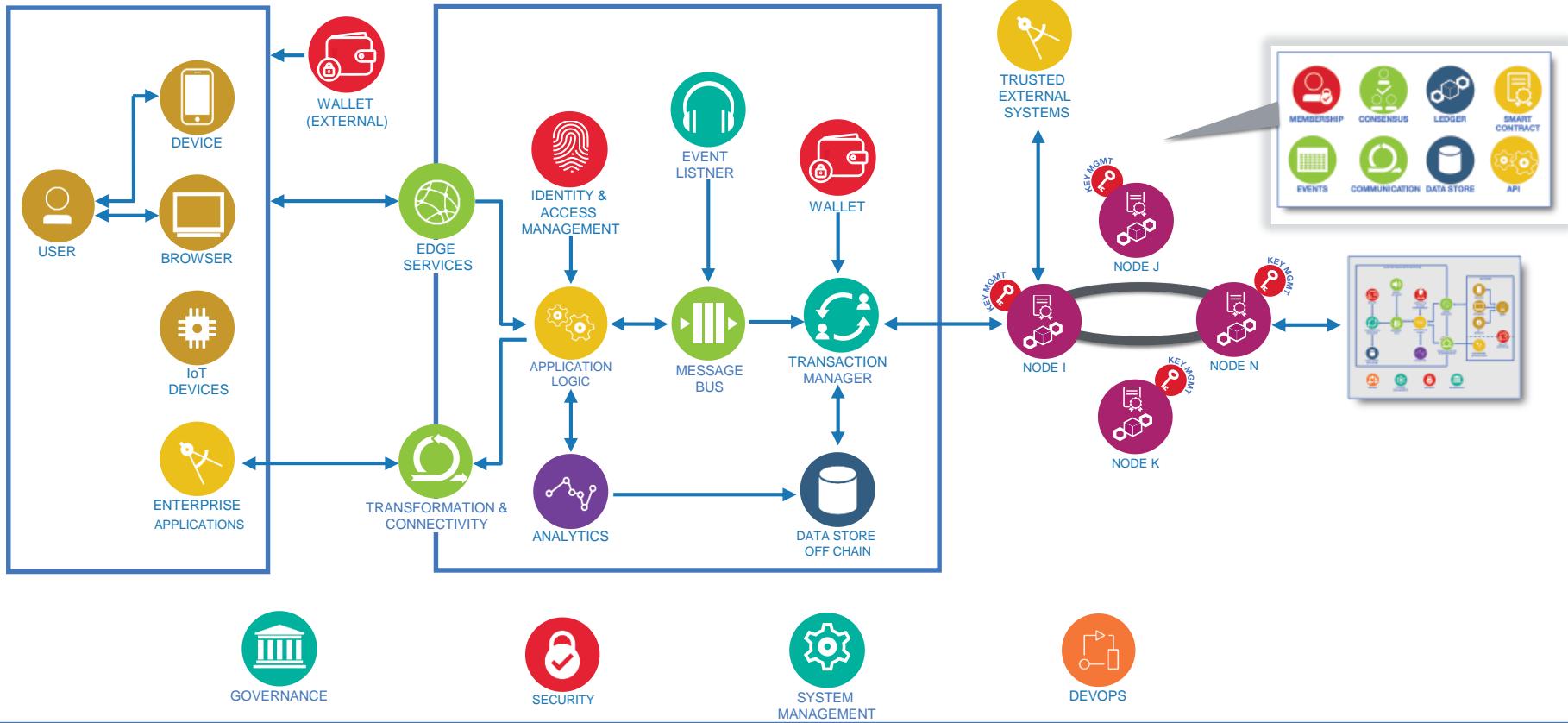


# Blockchain as a Software Component

把blockchain当做一个软件组件

- Complex, network-based software component
- Providing services
  - Data storage 1. 数据存储
  - Computation services 2. 计算服务
  - Communication services 3. 通讯服务
  - Asset management 4. 资产管理
- Features
  - Cryptographically-secure payment 机密安全的支付方式
  - Mining 可挖掘
  - Transaction Validation 支付确认
  - Incentive mechanism 奖励机制
  - Permission management 授权机制管理

# An Example Blockchain Reference Architecture



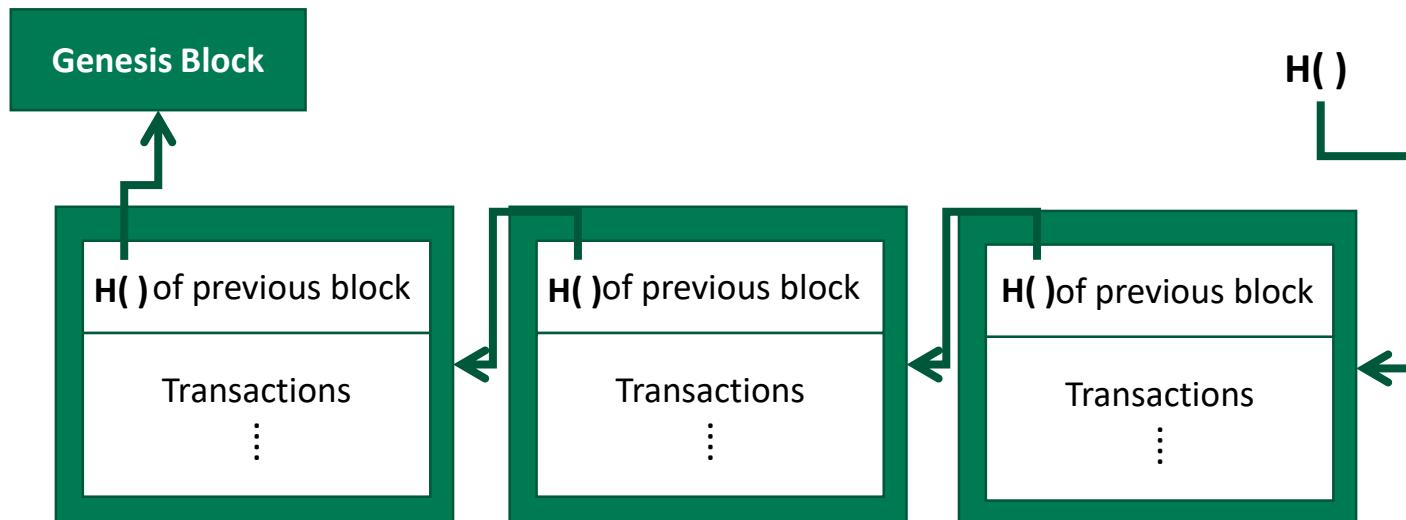
# Blockchain Component Service Options

- Data Storage
- Computation Service
- Communication Service
- Asset Management



# Blockchain as Storage Element

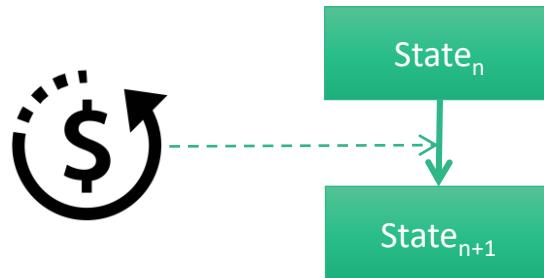
- Blockchain data structure
  - Linked list with hash pointers
- Tamper-proof
  - Computational constraints
  - Incentive scheme



# Blockchain as Storage Element

作为存贮单元,blockchain可以保证转账的完整性记录数据

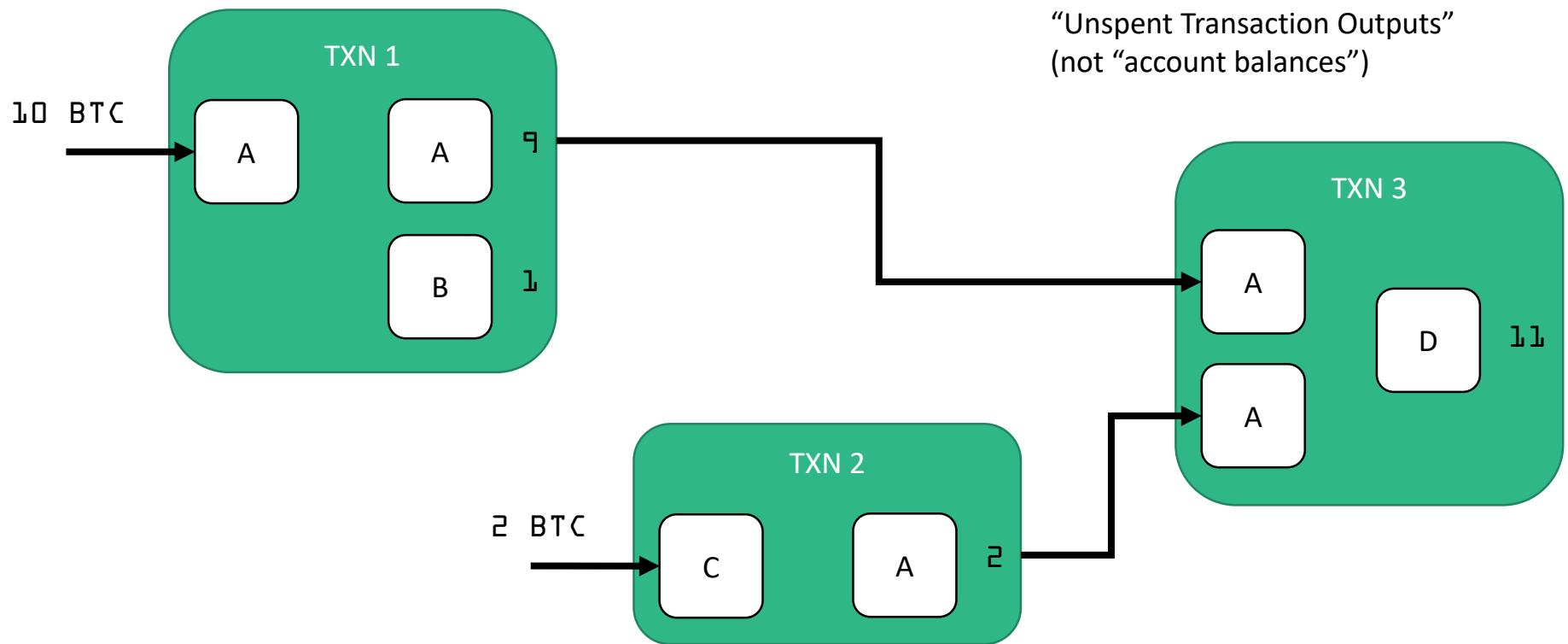
- Transaction Integrity
  - Transactions represent authorized state transitions
  - Transactions can record data
  - Transaction can transfer control of digital assets among participants
    - Crypto-currencies
    - Smart contract-based digital assets
  - Public key cryptography and digital signature are used to identify accounts
    - Ensure integrity and authorization of transactions initiated on a blockchain



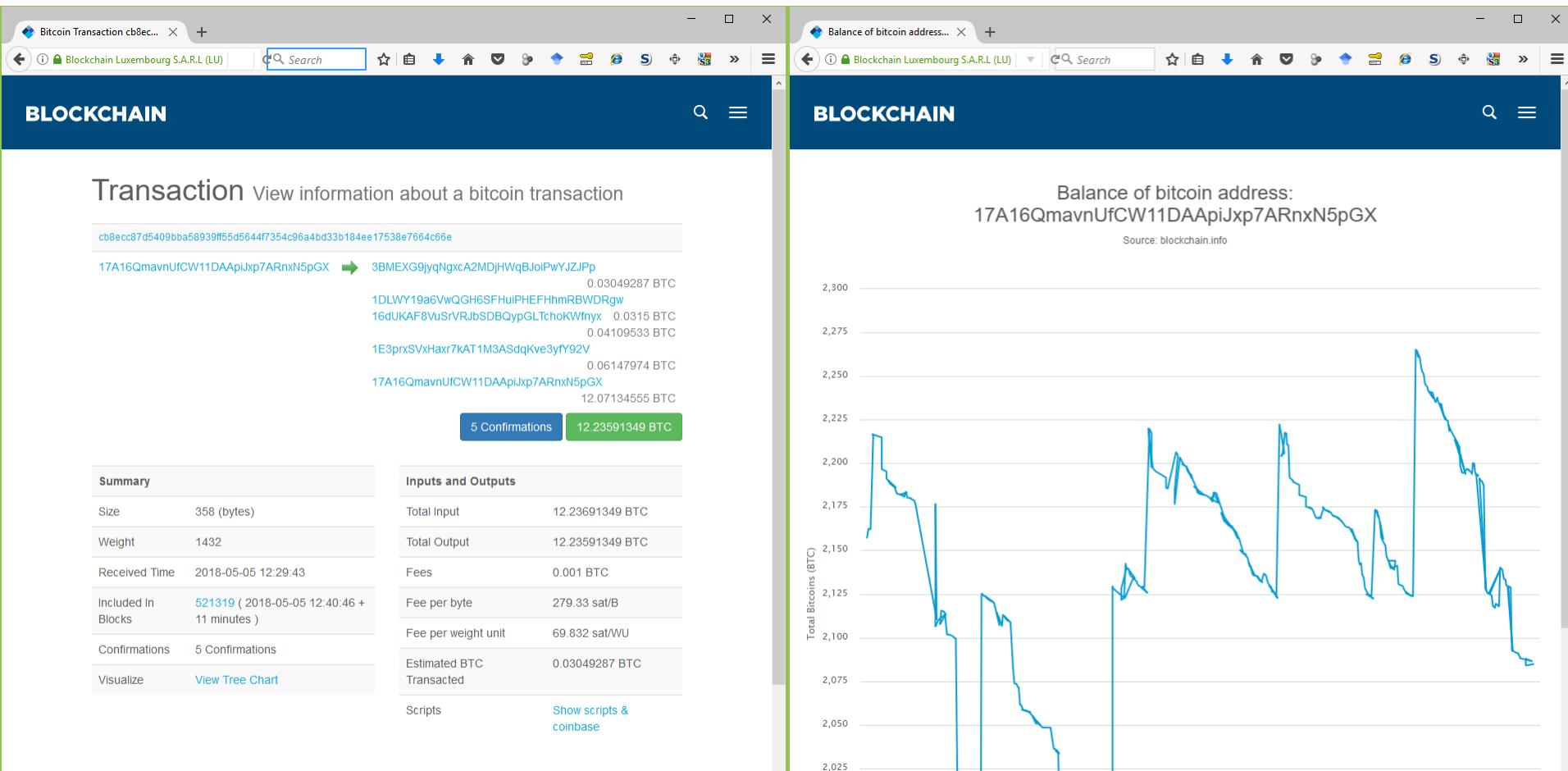
# Representation of Digital Assets

- Bitcoin Cryptocurrency
  - Collection of ~~unspent transaction outputs~~ (UTXO) from all previous transactions to that address
  - Tokens represented using conventions (e.g. tracking “color”)
- Ethereum
  - Cryptocurrency account balances in global system state
  - Tokens represented by smart contracts using storage

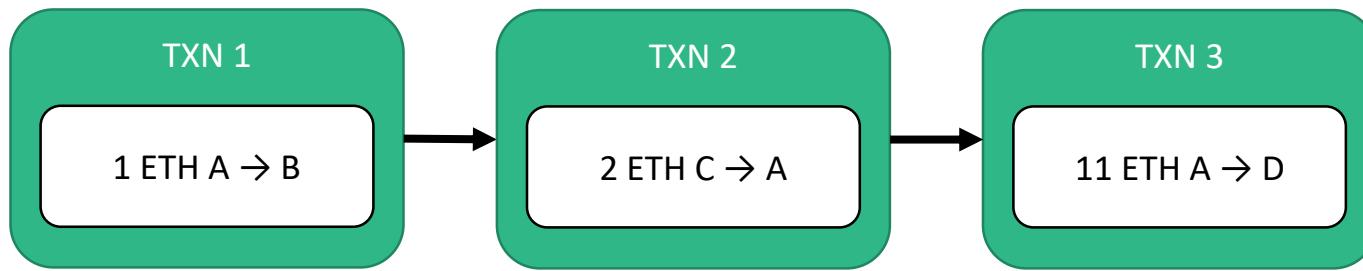
# Bitcoin Holdings – UTXO



# What Do Bitcoin Transactions/ Holdings Look Like?



# Ethereum Holdings – Accounts



Address	Balance
A	10
B	1
C	4

Address	Balance
A	9
B	2
C	4

Address	Balance
A	12
B	0
C	2

Address	Balance
A	1
B	0
C	4
D	11



# What Do Ethereum Transactions/ Holdings Look Like?

The image displays two side-by-side screenshots of the Etherscan blockchain explorer interface, illustrating the details of an Ethereum transaction and the holdings of a specific address.

**Left Screenshot (Transaction Details):**

- Header:** Ethereum Transaction 0xa14dee4df20ef4616cd3193d52b8c4abd38b1af38c63470545898ae47766d35a
- Breadcrumbs:** Home / Transactions / Transaction Information
- Sponsored Link:** Share Wi-Fi - earn cryptocurrency. Decentralized free Wi-Fi network. Get your bonus now. World Wi-Fi
- Overview Tab:** Selected. Shows the transaction information table below.
- Table (Transaction Information):**

Transaction Information		Tools & Utilities
TxHash:	0xa14dee4df20ef4616cd3193d52b8c4abd38b1af38c63470545898ae47766d35a	
TxReceipt Status:	Success	
Block Height:	5560602 (6 block confirmations)	
TimeStamp:	1 min ago (May-05-2018 01:04:01 PM +UTC)	
From:	0x6b0b7ecc2263aa562014b93b383e83111e6c84b	
To:	0x4c60fc9cdf2b334dc0c1ba821727c08d3da13db4	
Value:	5.076 Ether (\$4,175.21)	
Gas Limit:	31000	
Gas Used By Txn:	21000	
Gas Price:	0.000000005 Ether (5 Gwei)	
Actual Tx Cost/Fee:	0.000105 Ether (\$0.09)	
Nonce:	7	

**Right Screenshot (Address Details):**

- Header:** Ethereum Accounts, Addr... (0x4c60fc9cdf2b334dc0c1ba821727c08d3da13db4)
- Breadcrumbs:** Home / Accounts / Address
- Sponsored Link:** DocTailor - Legal Self Customisable Smart Contract Platform - Bridging the Gap Between Business & Cryptocurrency Holders - Join Now!
- Overview Tab:** Selected. Shows the address balance and transaction history.
- Table (Address Overview):**

Balance:	5.207054917052558449 Ether
USD Value:	\$4,281.14 (@ \$822.18/ETH)
Transactions:	121 txns
- Misc Tab:** Shows Address Watch and Token Balances.
- Table (Transactions):**

TxHash	Age	From	To	Value	[TxFee]
0xa14dee4df20ef461...	2 mins ago	0x6b0b7ecc2263aa...	IN	0x4c60fc9cdf2b334d...	5.076 Ether 0.000105
0xd155ec4c8c7bfbc...	4 hrs ago	0x4c60fc9cdf2b334d...	OUT	0x6b0b7ecc2263aa...	0.1 Ether 0.0001029

# Storing Arbitrary Data on Blockchain

- Two ways of storing data on blockchain
  - Adding data into transactions
    - Bitcoin
    - Ethereum
  - Adding data into contract storage
    - Smart contract on Ethereum
    - Smart contracts have an address, which is used to invoke the contract
    - Smart contract can only update its own storage
    - “Update” is only to the latest view of state (remember, append-only txns)
- Both ways store data through submitting transactions
  - Contain information of money transfer
  - Together with optional other data



# Comparison with Other Data Storage

- Comparison with Shared Centralised Database
- Comparison with Cloud Storage
- Comparison with Peer-to-Peer Data Storage
- Comparison with Replicated State Machines

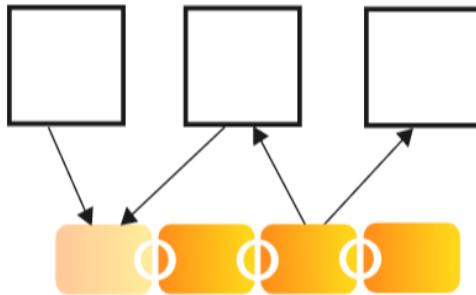


# NFPs: Blockchain vs Conventional Technology

- Non-Functional Property Trade-offs
  - (+) Integrity, Non-repudiation
  - (-) Confidentiality, Privacy
  - (-) Modifiability
  - (-) Throughput/ Scalability/ Big Data
  - (+ read/ - write) Availability/ Latency
- Many “limitations” are only for older public blockchains
  - e.g. hugely inefficient (more electricity than Ireland?)
  - e.g. very low performance (1 hour latency? max 3 tps?)

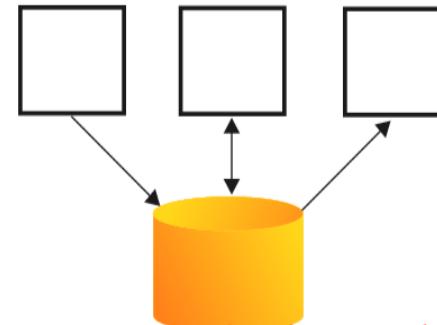


# Blockchain vs. Shared Centralised DB



交易只能append在后面

- Transactions are Append-Only
  - Mimic “CRUD” only as a view of the blockchain (transaction log)
- 多人达到共识,少数服从多数
  - Consensus by majority of peers agree on transactions
  - Usually no master, no trusted nodes



传统数据库啥都能干创建,读取,更新,删除

- Create, Read, Update, Delete
  - Log present, but only for admins
- Distributed Transactions (2 Phase Commit, Paxos)
  - Usually a master node, usually only a few trusted nodes

# Blockchain vs. Cloud

Blockchain	Cloud
No trusted single party	Cloud provider is trusted
Users can monitor or participate themselves as nodes that store the blockchain	Cloud provider store user data and provide access to the data
Data integrity is guaranteed (probabilistically)	Data integrity and access availability may not be guaranteed
No defined SLAs (service-level agreement) provided by public blockchain	Clearly defined SLAs

# Blockchain vs. Peer-to-Peer Data Storage

- Peer-to-peer Data Storage allow users to access data that is stored in other computers connected to the same peer-to-peer network
  - BitTorrent, IPFS (InterPlanetary File System)

Blockchain	Peer-to-Peer Data Storage
Very high availability <ul style="list-style-type: none"><li>• All nodes have the same shared copy of the blockchain data</li></ul>	Low availability for unpopular data
Not suitable for storing large data	Only store or distribute content the user wants to store or distribute
Strong data integrity	Hash pointer, checksum

# Blockchain vs. Replicated State Machines

	<b>Blockchain</b>	<b>Replicated State Machines</b>
<b>Fault Tolerance</b>	Use distribution to not depend on any single entity	<ul style="list-style-type: none"><li>• Replicate state at multiple servers</li><li>• Coordinate service requests from clients</li></ul>
<b>Consensus</b>	Ensure only one among multiple conflicting proposed transaction is included	Decide upon receiving update requests from components <ul style="list-style-type: none"><li>• Ensure only one client acquires a lock</li></ul>
<b>Voting</b>	large portion of the community to agree to achieve consensus	A quorum of voters with weighted votes
<b>Communication</b>	Transactions are replicated and persisted after it is included	<ul style="list-style-type: none"><li>• Transmitting state update data among components</li><li>• Information is replicated</li></ul>
<b>Facilitation</b>	Blocks and transactions are totally ordered	Requests are ordered

# Blockchain Component Service Options

- Data Storage
- **Computation Service**
- Communication Service
- Asset Management



# Data and Computational Platform

- **Blockchain 1.0 – Cryptocurrency**



第一代叫加密货币,第二代就升级成了加密经济状态机器主要是由于smart contract

- **Blockchain 2.0 – Cryptoeconomic State Machine**

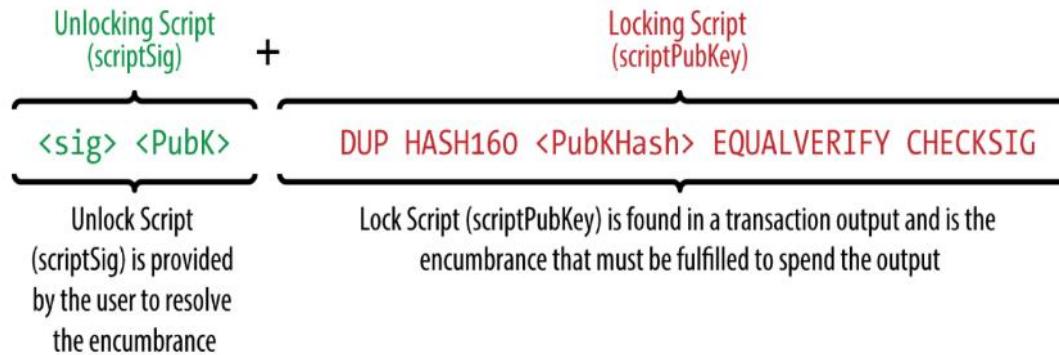
- Smart contract: Event-driven program (with state) that runs on a blockchain
  - Can realize more complex business logic
    - *More than simple currency/token-based value transfer*
    - Can represent digital assets on the ledger
  - Computational results stored on the public ledger



# Limited Computational Power of 1<sup>st</sup> Generation

最早 的smartcontract不支持复杂的逻辑控制流,

- Native smart contracts on Bitcoin do not support complex control flow definition



- External services allow end users to build self-executing contracts on Bitcoin
  - Executed by external Oracle
  - Integrity of execution is not guaranteed

# Turing-Complete 2<sup>nd</sup> Generation

- Ethereum is a general computational platform
  - Turing-complete programming language
  - In principle, as expressive as every other general purpose programming language (via Church-Turing thesis)
  - In practice, limitations on computational complexity
    - Gas limit
- Hyperledger Fabric, R3 Corda allow Java, etc
  - Can limit computation to nodes for parties of interest
  - Be careful to make your smart contracts deterministic

# 3<sup>rd</sup> Generation? Sub-Turing-Complete Again

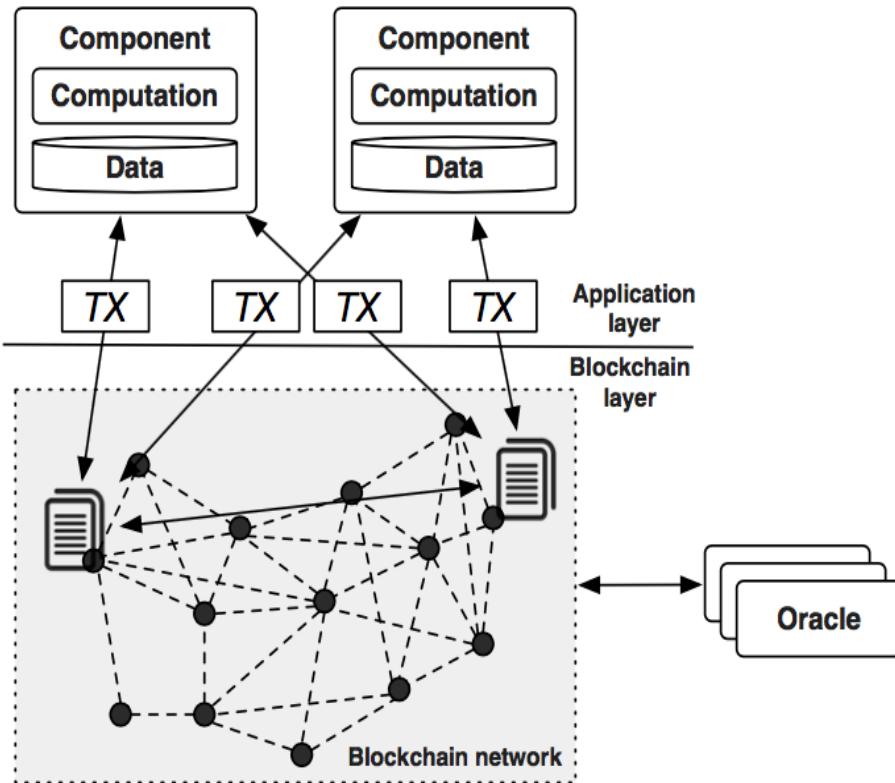
- A middle way – a somewhat expressive language, but still allowing automated formal verification
- e.g. Kadena's Pact language
  - Lists, datatypes
  - Functional: Conditionals, Fold, Map, Filter, ...
    - But not arbitrary recursion
  - Automatic verification using model checkers (Z3)
- e.g. DAML (Digital Asset)
  - Functional, non-Turing complete
  - Focus on workflow for exchange of rights & obligations
  - Tool support for formal verification

# Blockchain Component Service Options

- Data Storage
- Computation Service
- **Communication Service**
- Asset Management



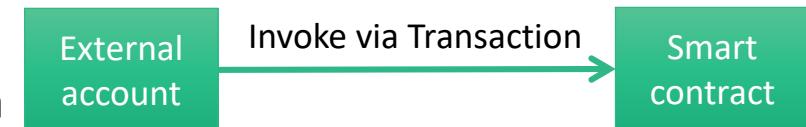
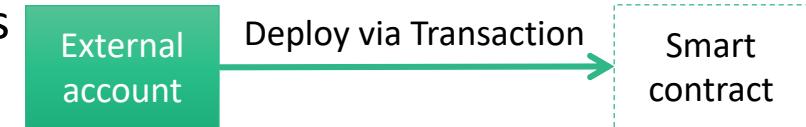
# Data Communication



- Components at application layer use blockchain as a mediator to transfer data
  - Sending data to blockchain using transactions
  - Query blockchain to retrieve data
- Blockchain provides data API
  - Access historical transactions
  - Filter historical transactions
- Local component monitoring update from new blocks
  - Relevant data in a local database

# Computation Communication 1/2

- Components use blockchain to coordinate computation
  - Submitting transactions to smart contracts to invoke their function
  - Using oracle to sign transactions depending on external state
- Typical control flow
  - Initiated from externally owned accounts
  - Transferred among contract accounts
- Contract termination
  - Cannot respond to transactions
  - Contract code remains on the blockchain
    - Permanently stored
    - In the creation transaction



# Computation Communication 2/2

- Oracle facilitates component coordination with external state
  - Some direct support for oracles
  - Others use oracles as external services
    - Interacting with blockchain through normal transactions
- Validation of transactions depends on external state
  - Platform-supported oracle can validate and sign transaction
    - Block transaction progress until the oracle completes
  - Oracle can be an external service injecting data into blockchain
    - Other smart contracts use that data to validate transaction
    - Delay is between the external state changes and time when those are recorded into blockchain
- Automated vs. human



# Blockchain Component Service Options

- Data Storage
- Computation Service
- Communication Service
- **Asset Management**



# Asset Management and Control Mechanism

- Native token of the 1<sup>st</sup> generation of blockchain
  - Cryptocurrency is the native asset
  - Identity of the cryptocurrency or associated data can represent other assets
    - Track claims of title over physical assets
    - Transactions record the transfer of title from one user to another
  - Limited due to small size of arbitrary data
    - **Bitcoin overlay network: colored coin**
      - Taint a subset of Bitcoin to represent and manage real-world assets
      - Few attributes can be recorded and few conditions can be checked within blockchain
- Smart contract of the 2<sup>nd</sup> generation of blockchain
  - Enable more expressive data structure
  - Flexibility for tokenizing a wider variety of assets



# Tokens as Digital Assets

TOKEN就类似于游戏中的货币,能够在游行中进行交换,给其赋予一  
值,就可以用他来在游戏当中买东西,转账(这是完全适应游戏中的环  
应为现实中的钱在里面其实用不了,也不可能刷信用卡)

- What is a Token?
  - Like plastic discs/boarding pass/ticket/ in the physical world
  - Digital asset on the blockchain – current “owner” etc can be checked by looking at ledger
- A token represents a bundle of rights and obligations
  - Can represent digital assets or physical assets
    - Fungible: interchangeable, like cryptocurrencies, gasoline
    - Non-fungible: Unique and cannot be interchanged, like cryptokitties, artwork and land
  - Transferrable (or not); Reusable (or not); Exclusive (or not); ...
- Tokens for “ownership” of assets?
  - “Property” is a collection of rights
  - Can the blockchain legally bind ownership or change of ownership?
    - You could self-impose contractual conditions on token and underlying property, to try to ensure the blockchain record is consistent with legal ownership
    - But, contracts might not survive bankruptcy; and Courts can order change of ownership without the blockchain



# Digital Assets and Blockchains: Symbiosis

- The blockchain ensures digital assets are not replicated
  - For digital information, you can make identical copies
  - For money and other assets, don't make identical copies!
  - The global public ledger means everyone can check



- Digital assets (esp. cryptocurrency for public blockchains) ensure the blockchain is operated
  - Provide incentives for nodes to operate the blockchain
    - Mining reward claimed under a convention, by the node that creates a block
    - Transaction fees – offered by the transacting party to the node which includes the transaction in a block

# Token Standards on Ethereum

- ERC20 for fungible tokens
- ERC841 for non-fungible tokens

```
1 // -----
2 // ERC Token Standard #20 Interface
3 // https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md
4 //
5 contract ERC20Interface {
6     function totalSupply() public view returns (uint);
7     function balanceOf(address tokenOwner) public view returns (uint balance);
8     function allowance(address tokenOwner, address spender) public view returns (uint remaining);
9     function transfer(address to, uint tokens) public returns (bool success);
10    function approve(address spender, uint tokens) public returns (bool success);
11    function transferFrom(address from, address to, uint tokens) public returns (bool success);
12
13    event Transfer(address indexed from, address indexed to, uint tokens);
14    event Approval(address indexed tokenOwner, address indexed spender, uint tokens);
15 }
```

# Design and Trade-offs in Blockchain Applications

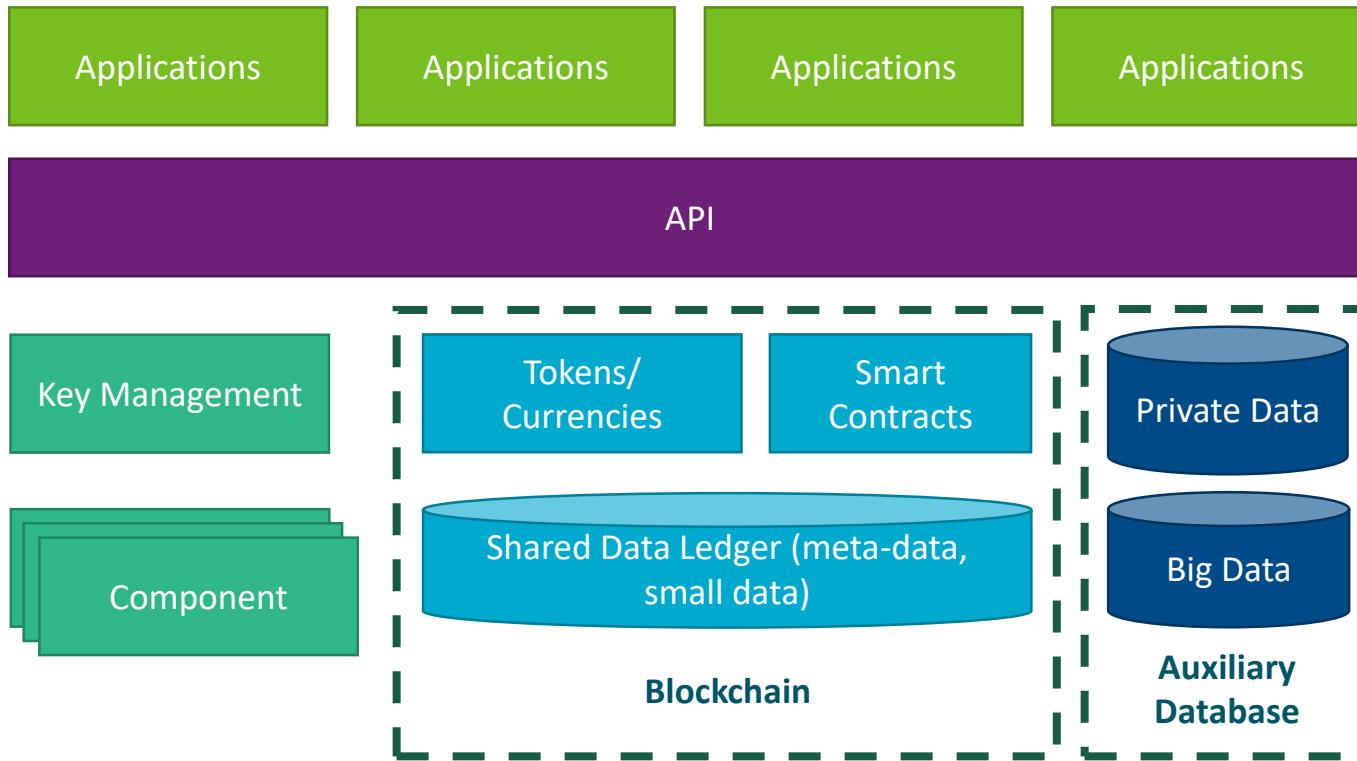


# Use Blockchain for What It's Good For

- Trustworthy and efficient ways to work together
  - Focus on spaces between individuals, organisations
    - Data integrity for information sharing
    - Neutral ground for process coordination
  - Logically centralises information
  - Administratively decentralises control
- Digital Assets and Tokens
  - Authorised by issuer, transferrable between cryptographic IDs
  - Representing physical assets, digital assets, services, rights, ...
  - Be careful about legal title for property: blockchain is not magic



# Recall: Blockchain in Larger Software System



# How to use blockchains in a design?

- Later lectures covering topics in design process, blockchain design patterns, etc
- Some themes
  - Choice and configuration of blockchain
    - Private (Consortium) vs. Public 公链私链
    - Single logical chain vs. network of distributed ledgers
  - Use programming design patterns for smart contracts
  - Combine on-chain with off-chain components
    - Hashed content on-chain (content off-chain) 只把hash值放在链上, content放在链下
    - Signed or encrypted content on-chain (keys off-chain) 签名的content链上, keys链下
    - “State channels” judges on-chain (computation & comms off-chain)
- Today, some illustrative examples

# 3 Examples from Data61/Treasury Reports

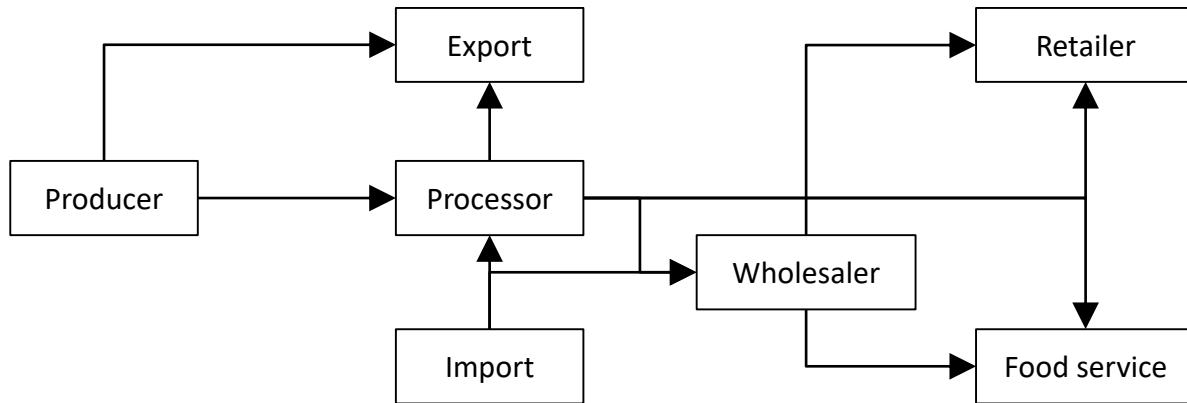


- <http://www.data61.csiro.au/blockchain>
- *Risks and Opportunities for Systems Using Blockchain and Smart Contracts*
- What are technical risks & opportunities for use cases?

# Potential Use Cases

- Financial Services
  - Digital currency
  - (International) payments
  - Reconciliation
  - Settlement
  - Markets
  - Trade finance
- Government Services
  - Registry & Identity
  - Grants & Social Security
  - Quota management
  - Taxation
- Enterprise and Industry
  - Supply chain
  - IoT
  - Metered access
  - Digital rights & IP
  - Data management
  - Attestation
  - Inter-divisional accounting
  - Corporate Affairs
- Three Illustrative Cases Selected
  1. Agricultural supply chain
  2. Open data registry
  3. Remittance payments

# Agricultural Supply Chain – Use Case

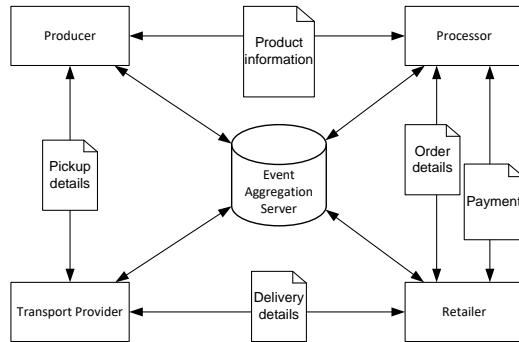


Interoperability  
Latency  
Integrity  
Confidentiality  
Scalability

# Agricultural Supply Chain – Designs

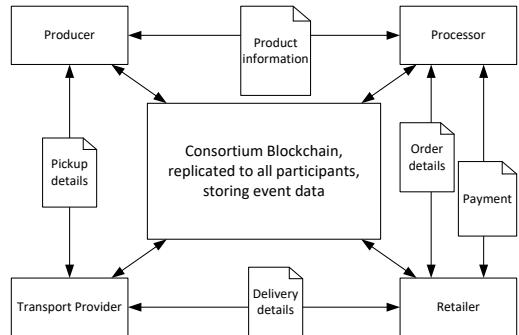
## 1. Conventional

Point-to-point messaging and event aggregation server

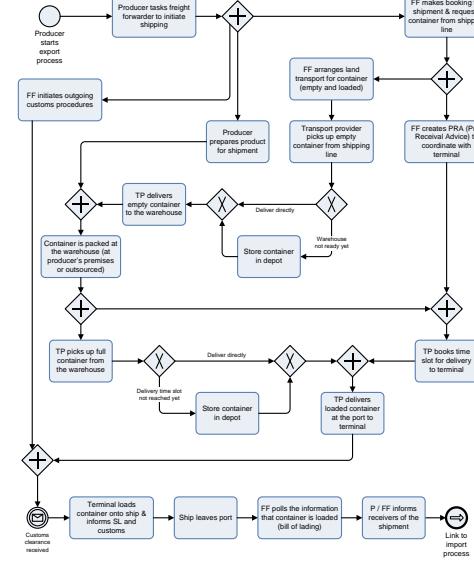


## 2. Event Tracking on Blockchain

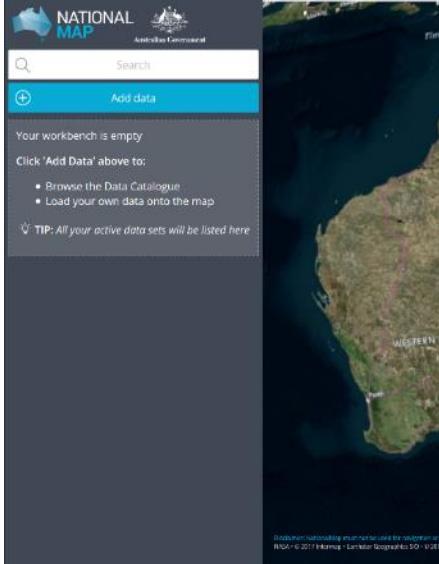
Point-to-point messaging and event aggregation on blockchain



## 3. Supply chain process coordination on blockchain as smart contracts



# Open Data Registries – Use Case



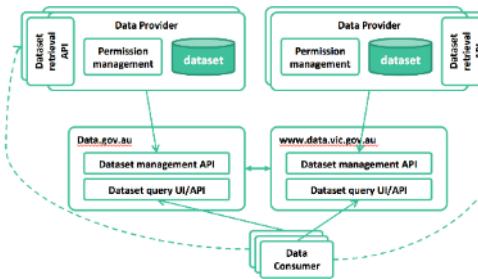
A screenshot of the data.gov.au website. The header features the Australian Government logo and the data.gov.au logo. The main navigation menu includes Datasets, Organisations, About, Site Statistics, Request Data, Use Cases, National Map, and Toolkit. Below the menu is a search bar with the placeholder 'E.g. environment'. A sidebar on the left shows a message about an empty workbench and instructions to add data. The central content area contains a section titled 'Search data' with a search bar, popular tags like Earth Sciences and GA Publication, and a statistics box showing 35.8k datasets, 7k API enabled resources, 24.7k openly licenced datasets, and 25 unpublished datasets. There are also sections for 'Latest data.gov.au News' and 'Guest Post: Help to shape ASIC Registry's public datasets'.

Integrity  
Availability  
Read Latency  
Interoperability  
Barriers to access

# Open Data Registries – Designs

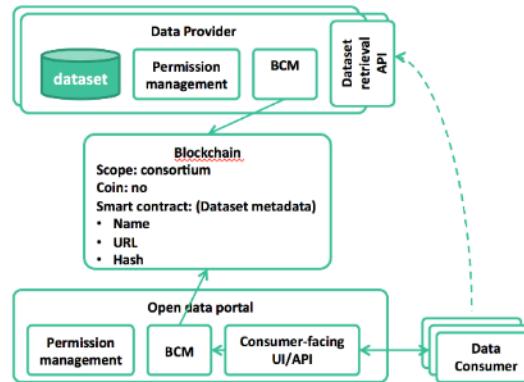
## 1. Conventional

Registry operated by single agency



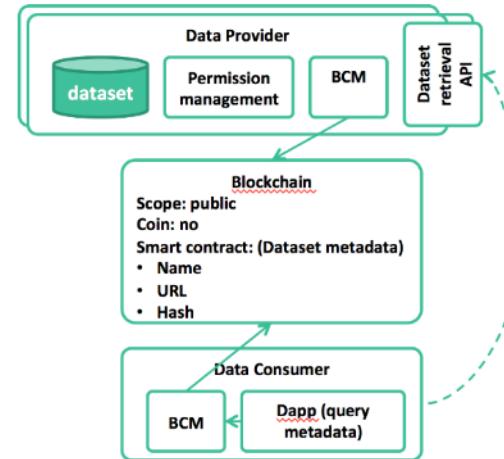
## 2. Consortium across data providers

Public access still controlled through a portal

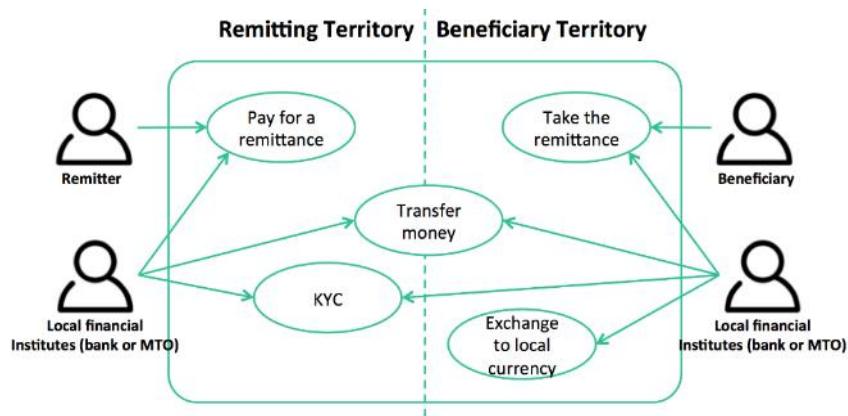


## 3. Registry on public blockchain

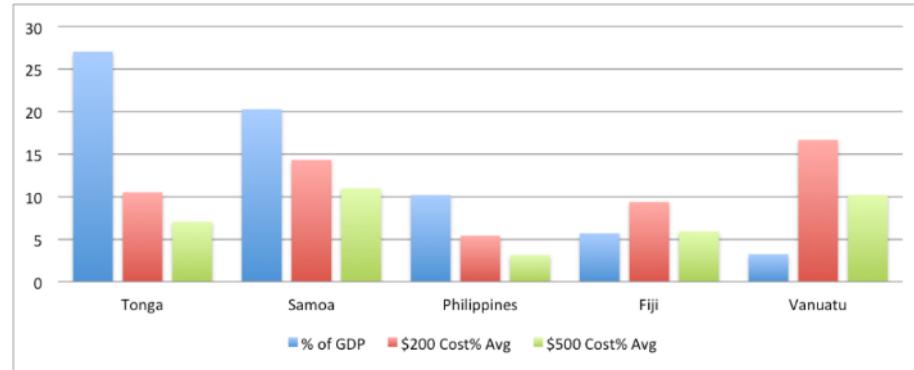
Agency only controls entries included on official index



# Remittance Payments – Use Case



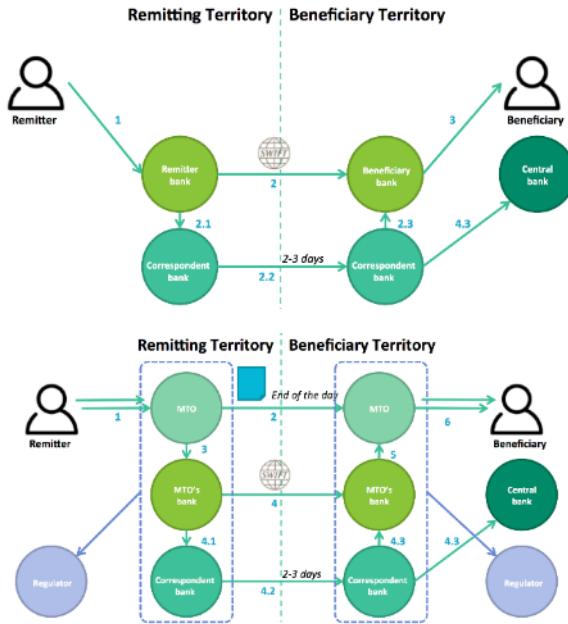
Write Latency  
Cost  
Cost transparency  
Controlled confidentiality  
Low barriers to entry



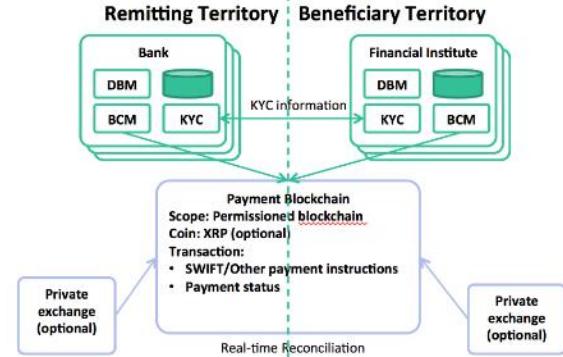
# Remittance Payments – Designs

## 1. Conventional

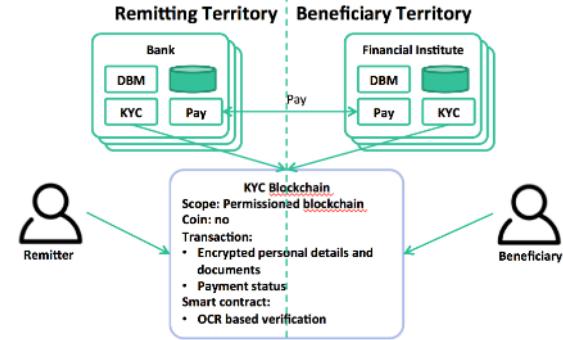
Through bank or MTO



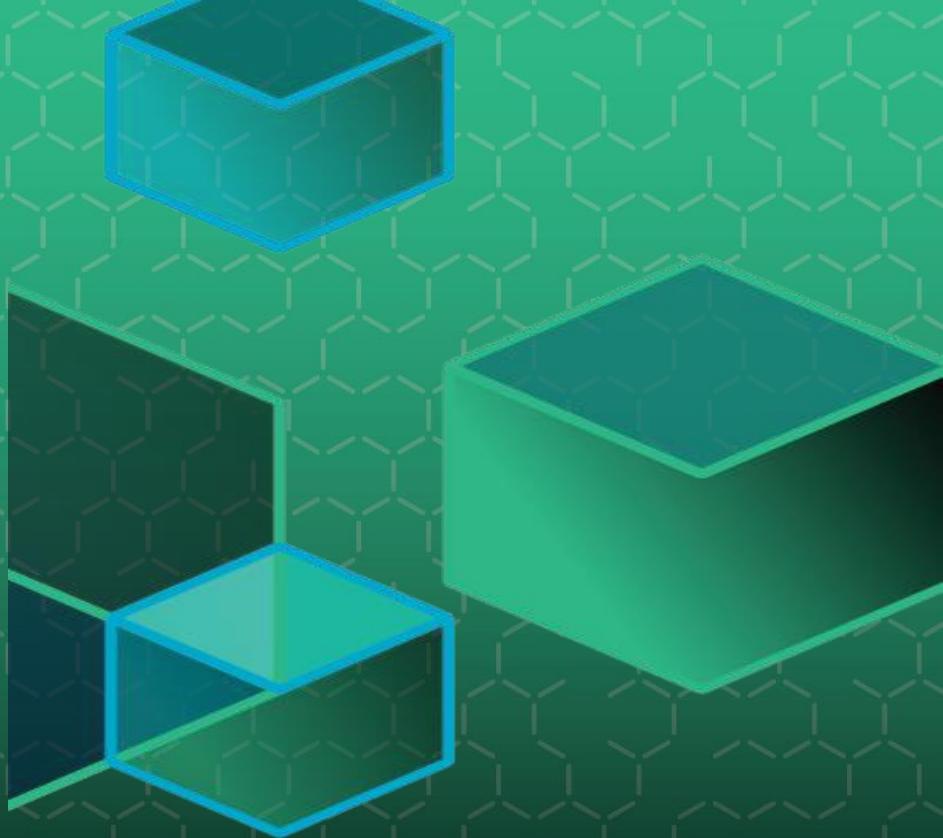
## 2. Payment through blockchain



## 3. KYC through blockchain



# Summary



# Summary

- What is Software Architecture?
  - Architectural design choices impact Non-Functional Properties (NFPs)
  - Use Architectural Models (Views and Viewpoints)
    - To communicate and analyse systems
  - Analyse NFPs and consider NFP Trade-offs
- Blockchain in Software Architecture
  - Blockchain as Component, Connector, Configuration
  - Blockchain Component Services
    - Data Storage
    - Computation Service
    - Communication Service
    - Asset Management
- Design and Trade-offs in Blockchain-based Applications





# THANK YOU

[www.data61.csiro.au](http://www.data61.csiro.au)