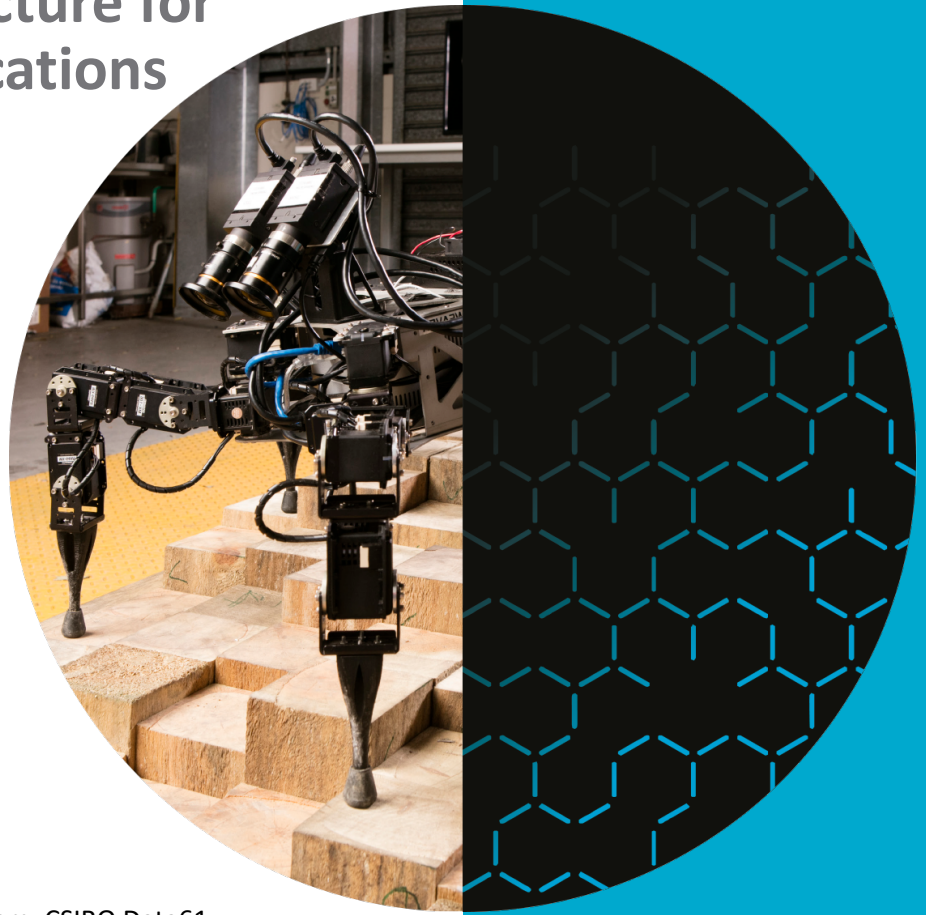
COMP6452
Software Architecture for Blockchain Applications



Blockchain in Software Architecture

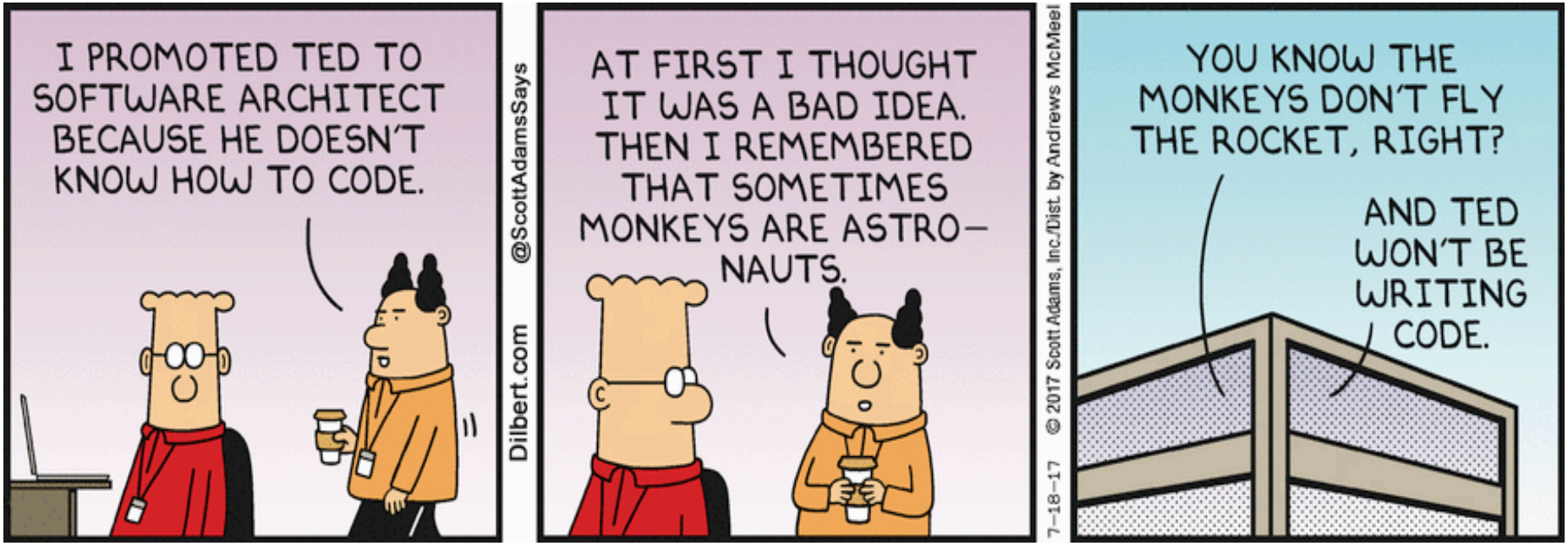
Xiwei (Sherry) Xu

| Senior Research Scientist @ AAP team, CSIRO Data61
| Xiwei.Xu@data61.csiro.au

Outline

- Software Architecture Basics
- Blockchain in Software Architecture

Software Architecture Basics



The good thing about bubbles and arrows, as opposed to programs, is that they never crash. ”

– Bertrand Meyer

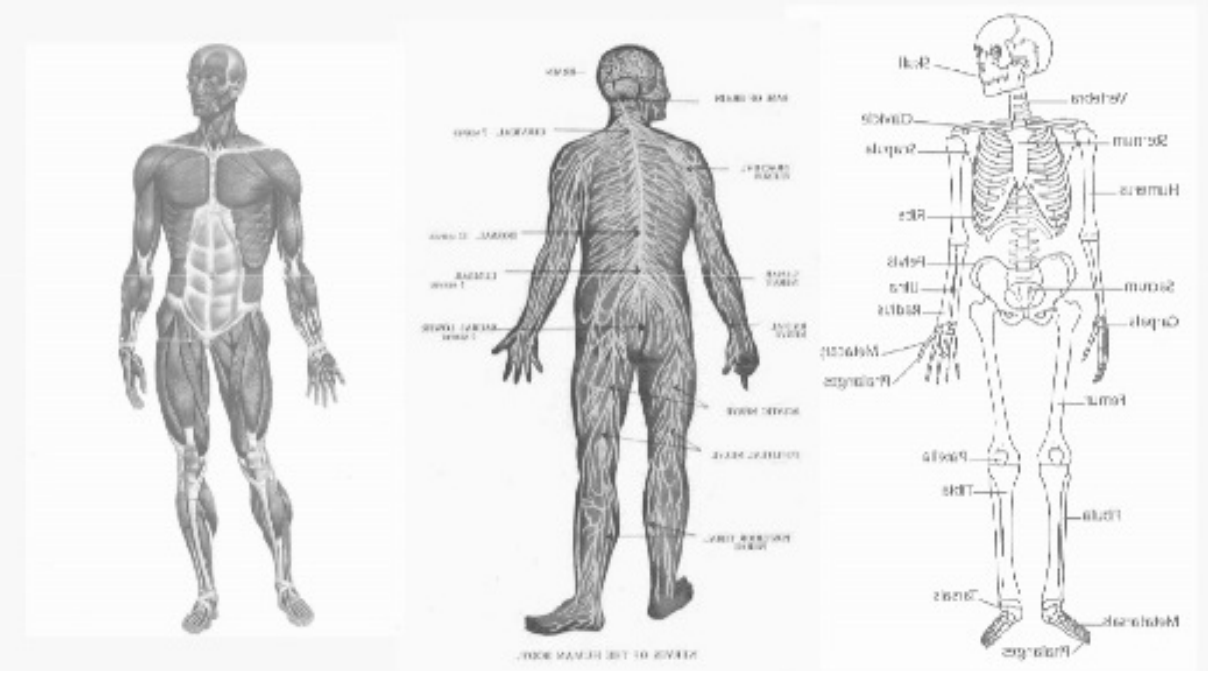
Source: http://dilbert.com/strip/2017-07-18

Architecture is an Abstraction

- 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
    - c.f. “marketecture”
- “All models are wrong, but some models are useful”

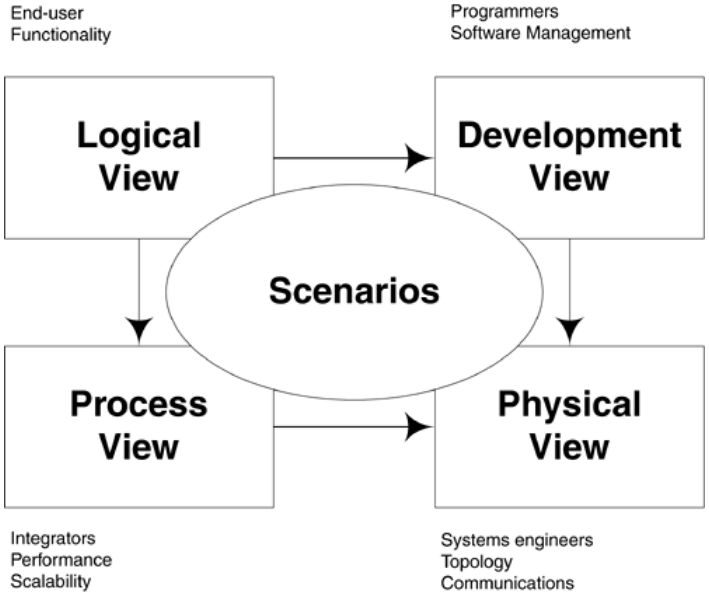
  - George Box
- Discussion: Why Abstraction?
- Models are abstractions; what to model and what to abstract depends on the purpose
  - “[A] model which took account of all the variegation of reality would be of no more use than a map at the scale of one to one” - Joan Robinson, 1962

Viewpoints and Views

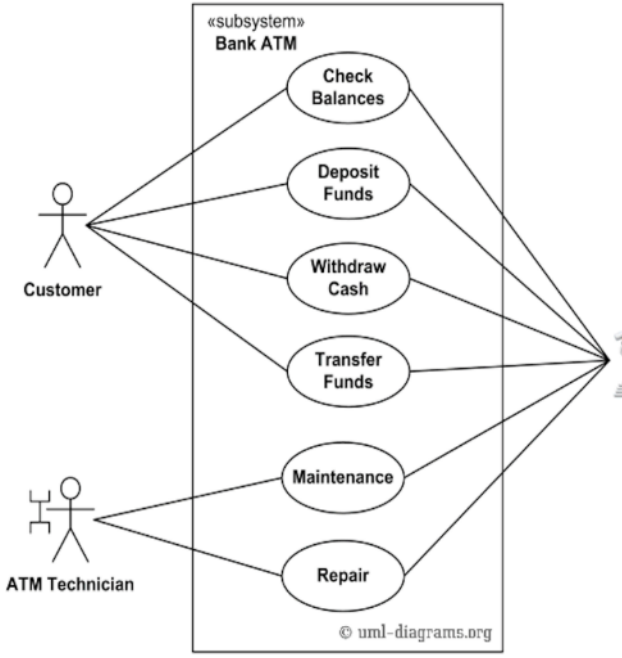


Krutchen's 4+1 View Model

- **Logical**: architecturally-significant elements and the relationships between them.
- **Process**: concurrency and communications elements
- **Physical**: how the major processes and components are mapped to application's 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

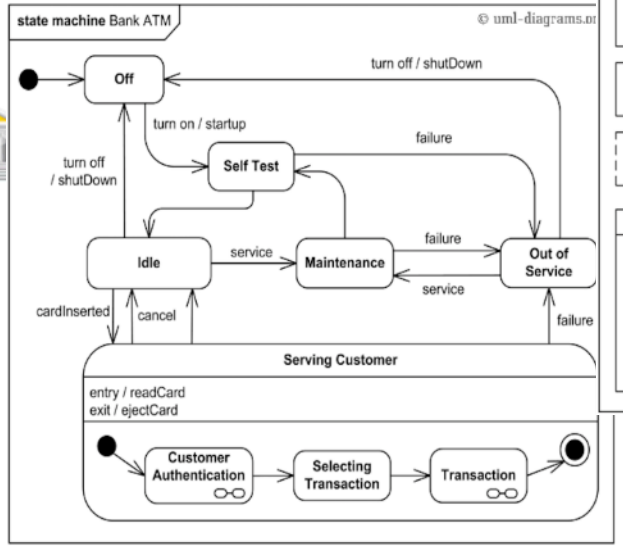


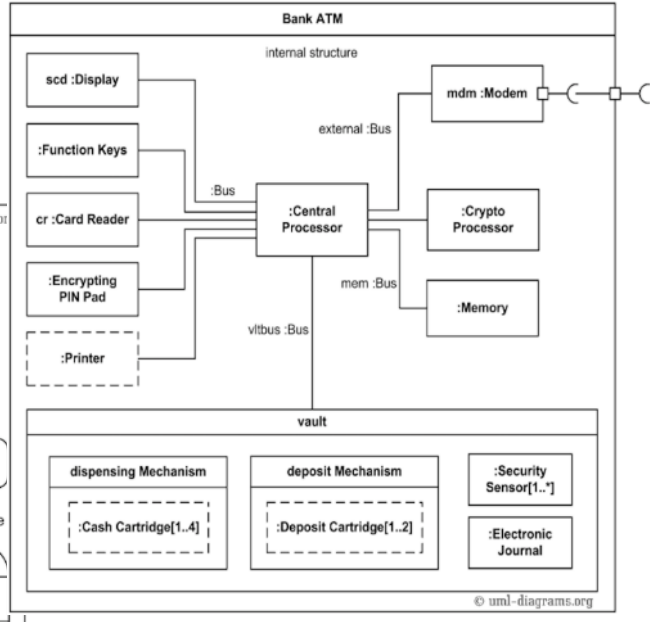
UML as Viewpoints and Views



UML Use Case

UML Statechart

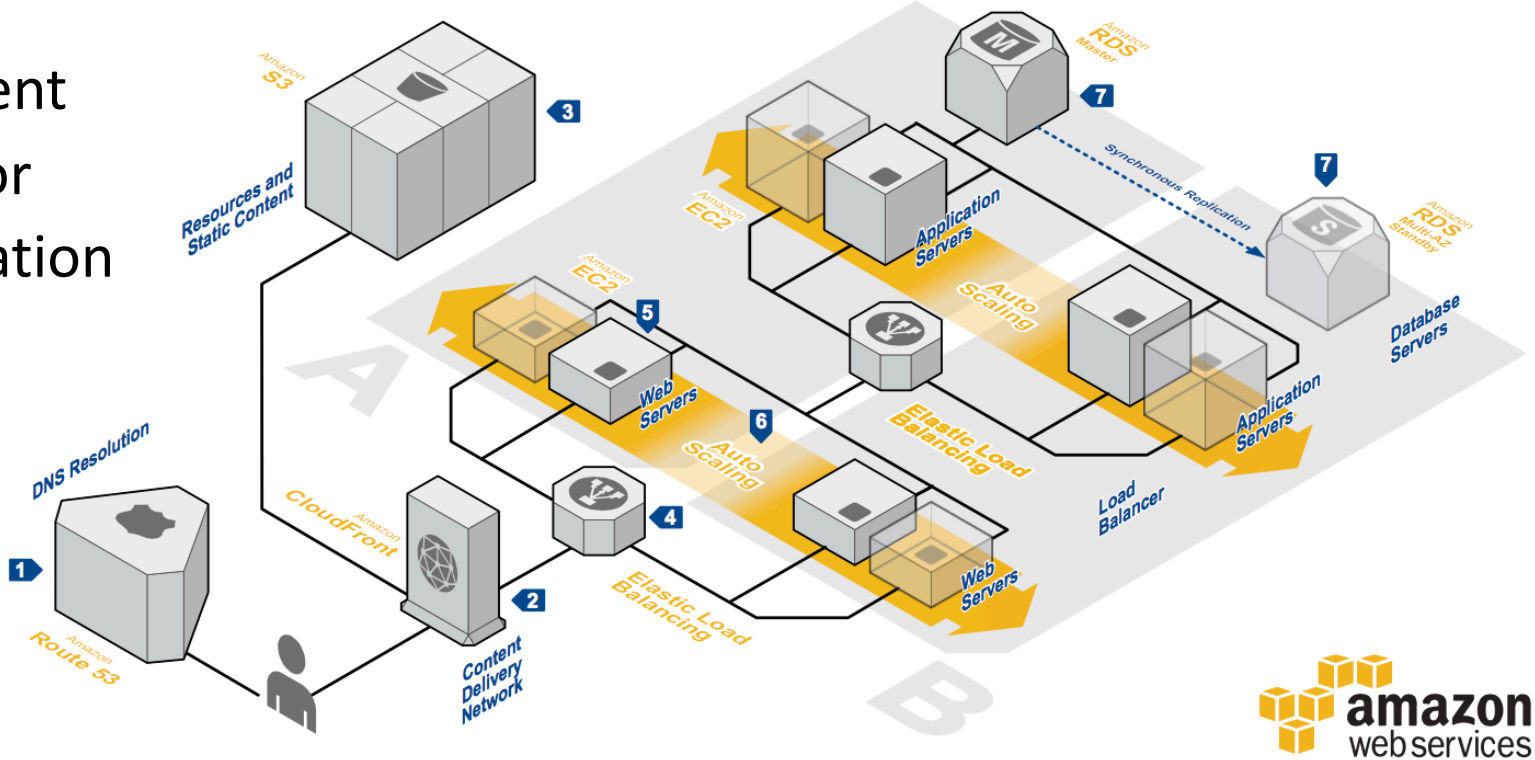




UML Composite Structure

Software Architecture's Element

- Component
- Connector
- 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

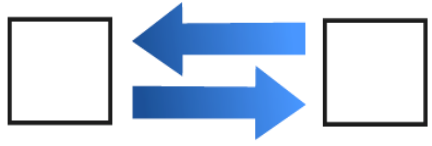
Software Connectors

- 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

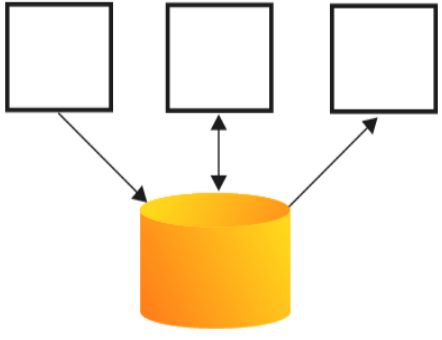
Example Software Connectors

File Transfer

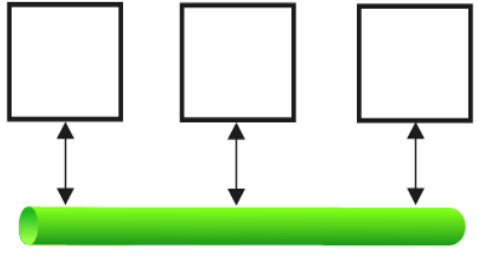
Stream



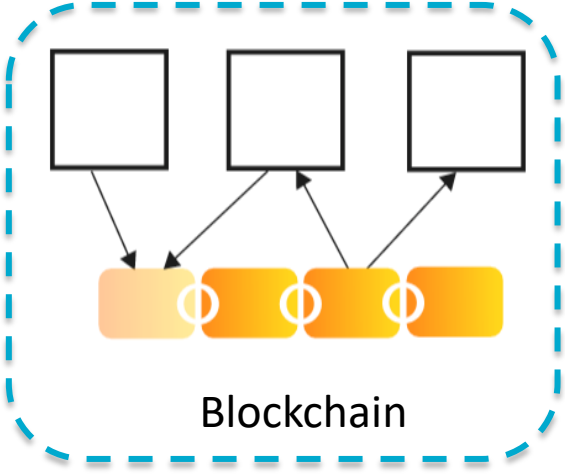
Remote Procedure Call



Shared Database



Message Bus



Blockchain

Architecture Specifies Component Communication

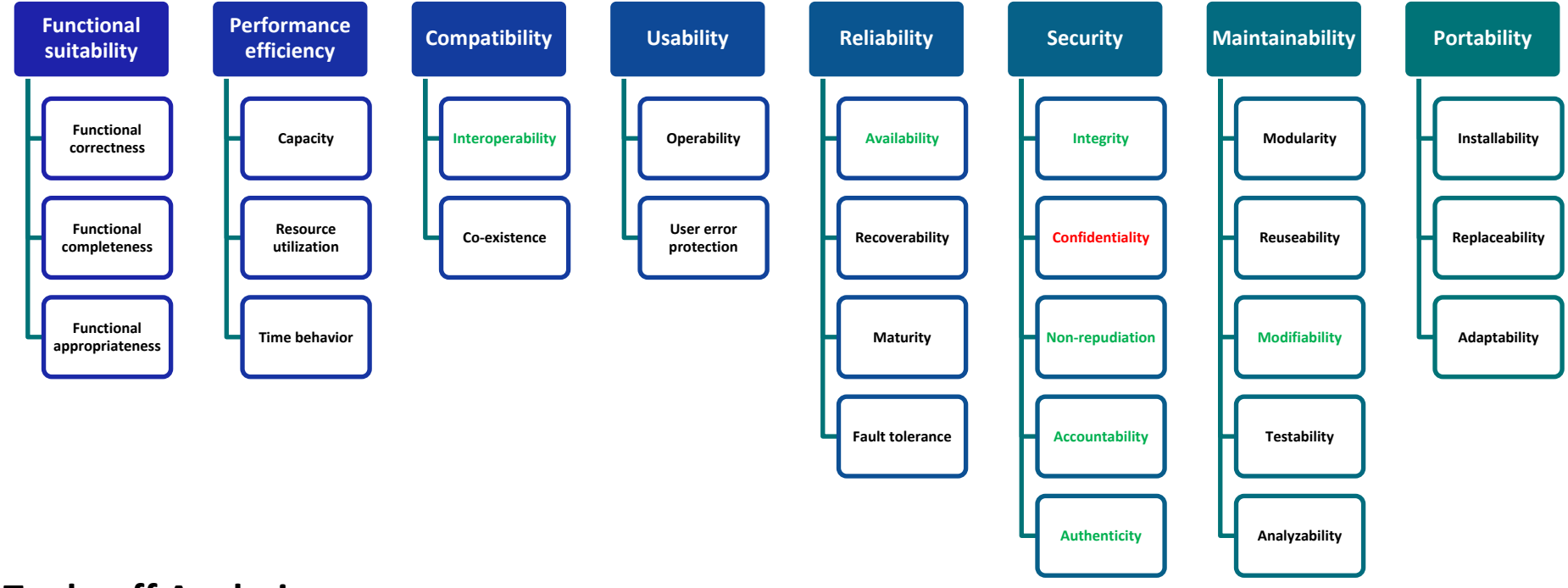
- Data passing mechanisms, e.g.:
  - Function call
  - Remote method invocation
  - Asynchronous message
- Control flow
  - Flow of messages between components
  - Sequential
  - Concurrent/parallel
  - Synchronization

Non-Functional Properties

Non-Functional Properties arise from Architectural Design Choices

- There are two kinds of requirements:
  1. Functional Requirements (i.e. what are the inputs and outputs)
  2. Non-Functional Requirements (a.k.a. *Qualities*, or *-illities*)
     - e.g. “Performance” (latency, throughput, …)
     - e.g. “Security” (confidentiality, integrity, availability, privacy, …)
     - e.g. Usability, Reliability, Modifiability, …

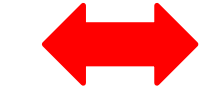
ISO/IEC 25010:2011 Quality Model



Trade-off Analysis

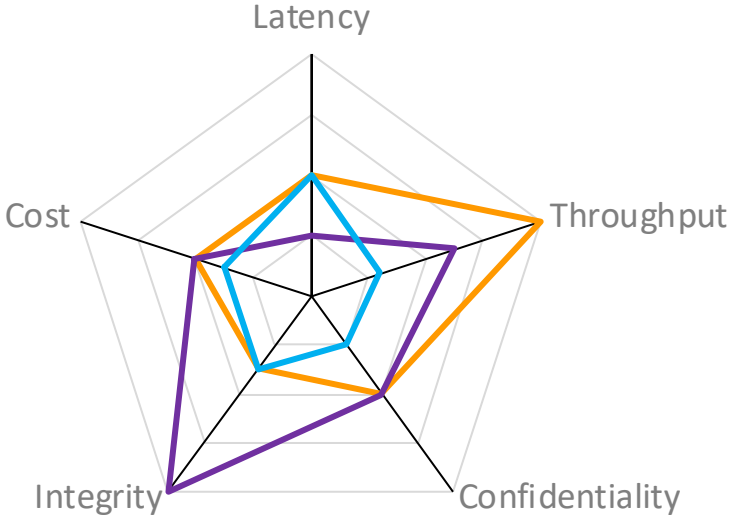
Why Non-Functional Properties Matter

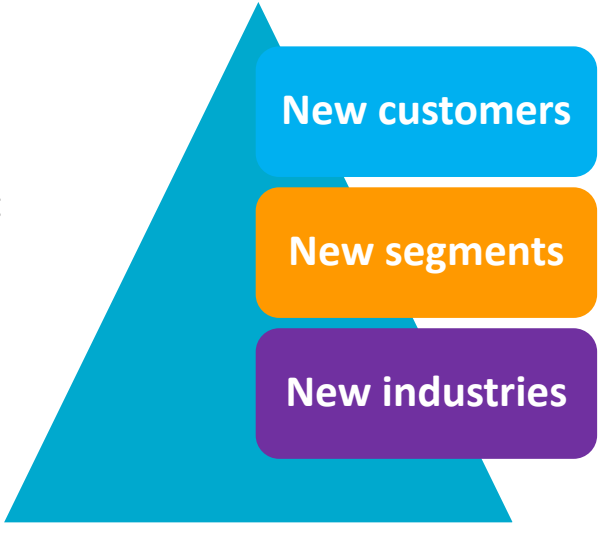
Regulatory Requirements



System Performance

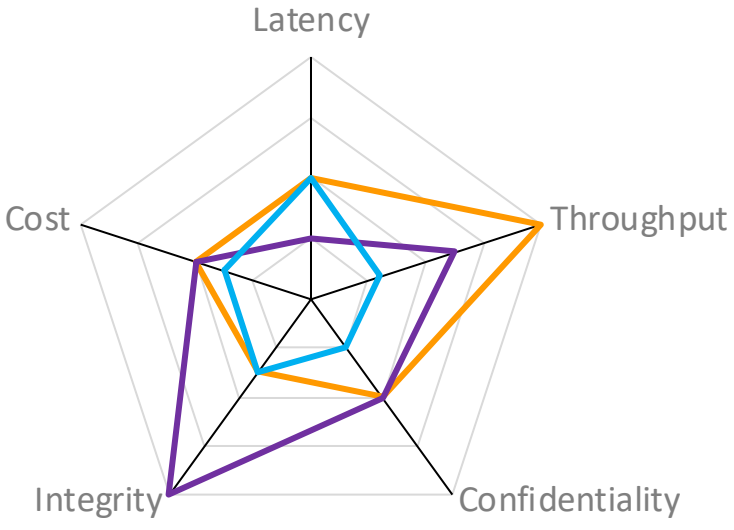
Market Opportunities

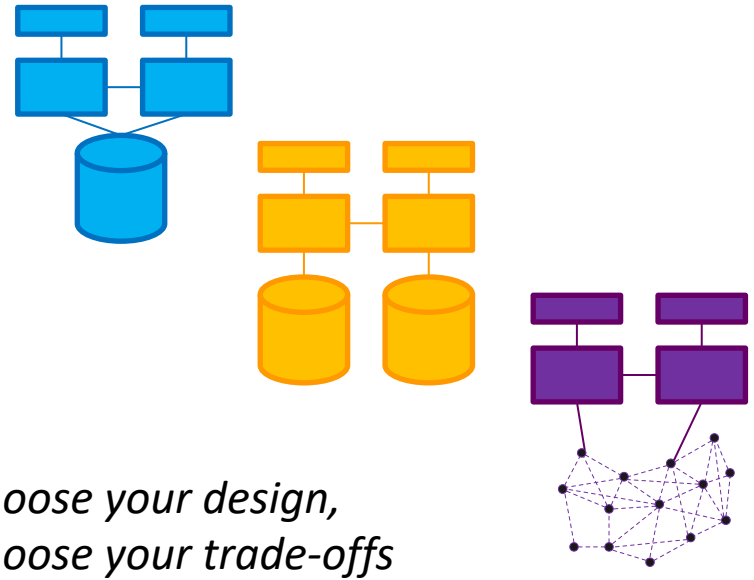




Software Architecture

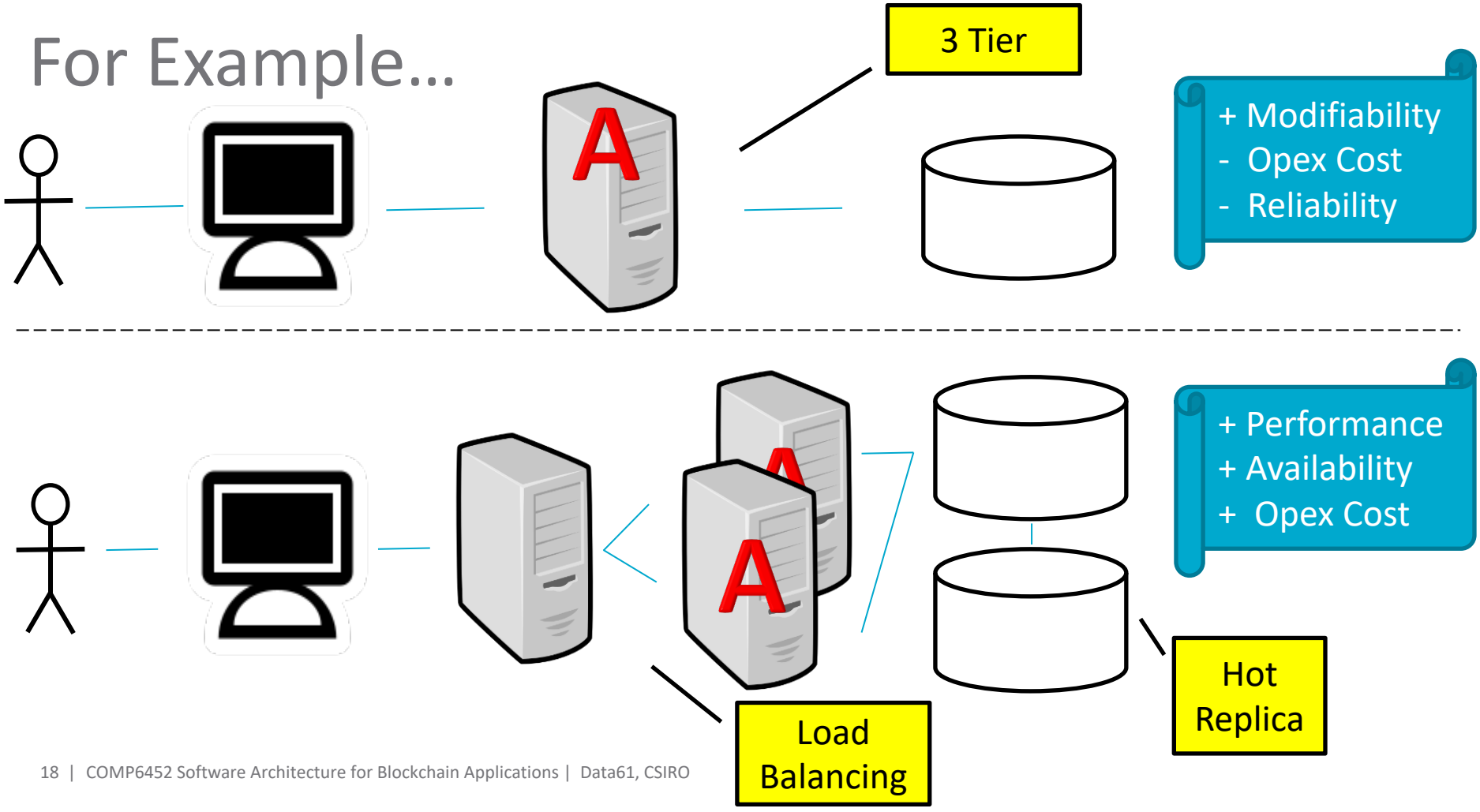
Non-Functional Properties arise from Architectural Design Choices



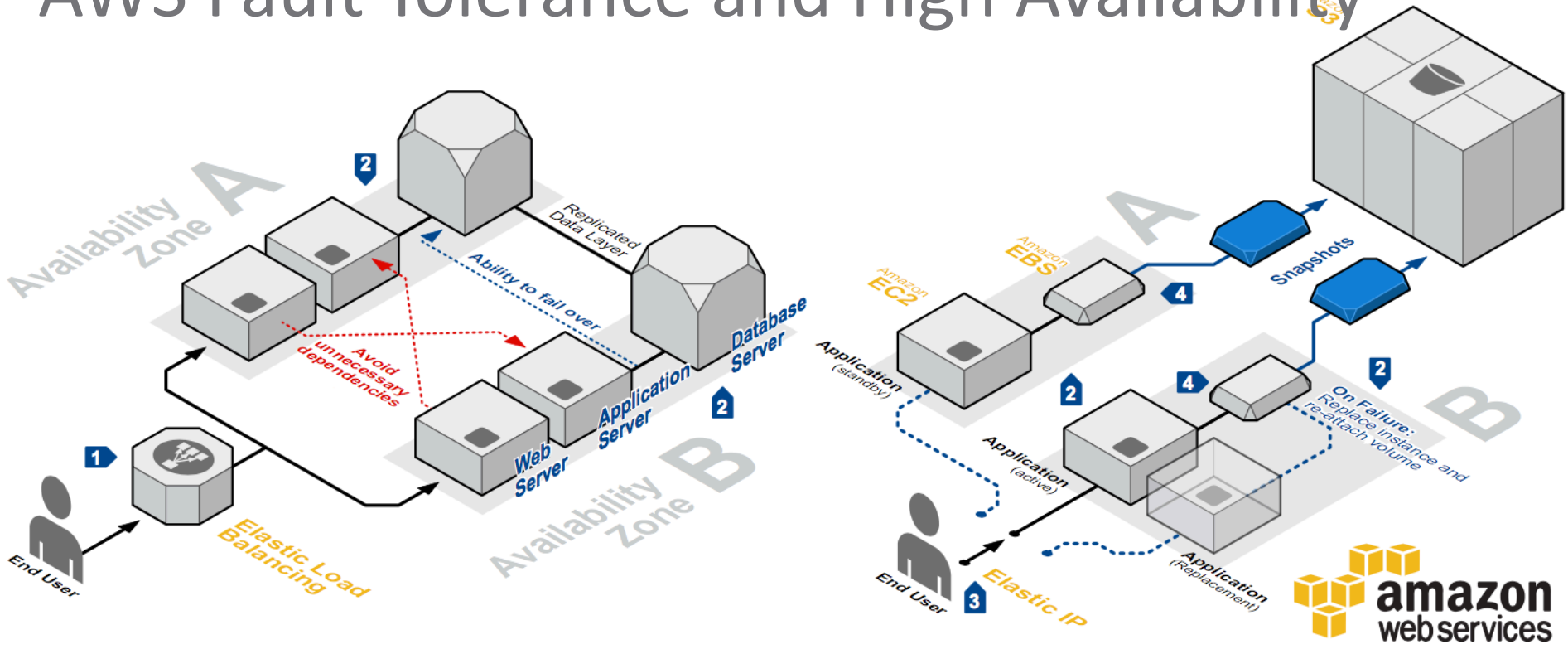


Choose your design,
Choose your trade-offs

For Example…


3 Tier
+ Modifiability
- Opex Cost
- Reliability
+ Performance
+ Availability
+ Opex Cost
Load Balancing
Hot Replica

AWS Fault Tolerance and High Availability

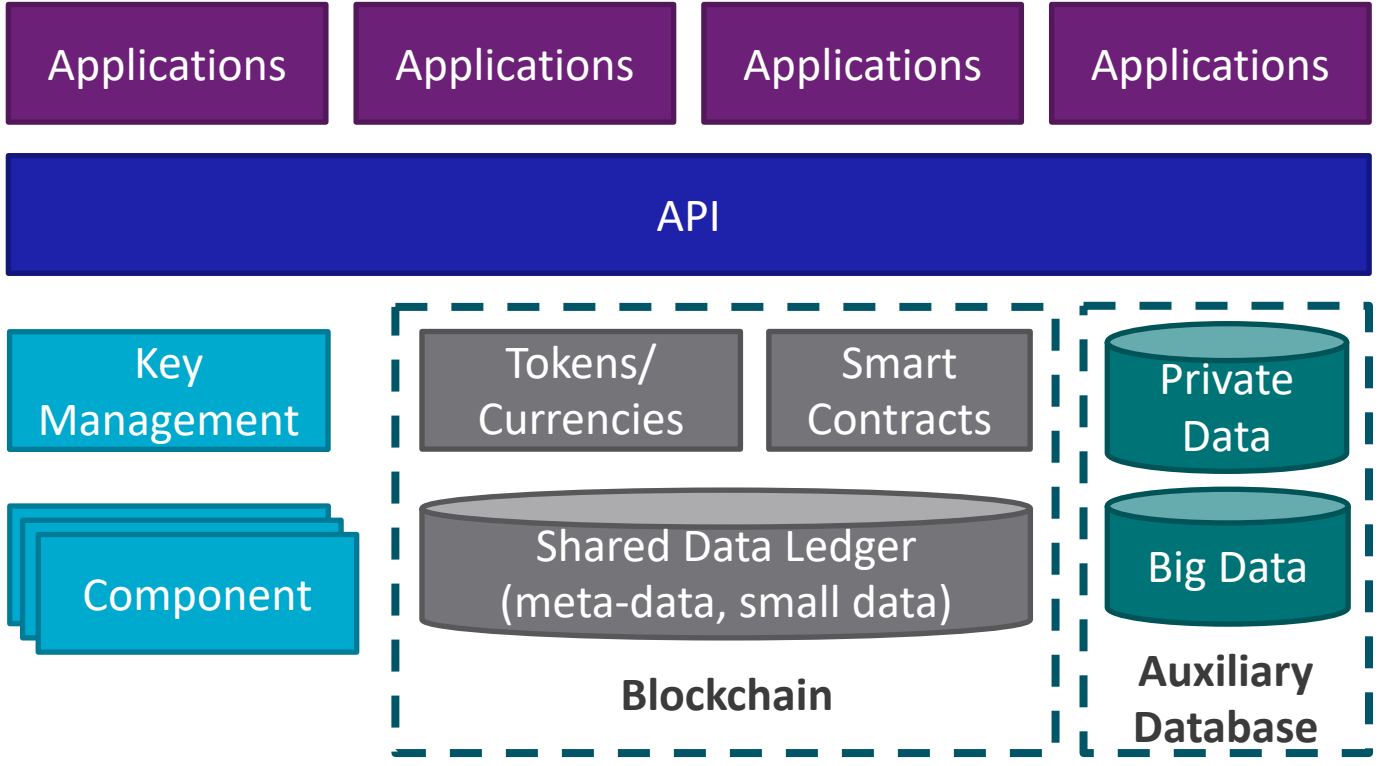


Student Task

- Tell us about your experiences with software architecture

Blockchain in Software Architecture

Blockchain in Larger Software System



Benefits

- Understanding important architectural impacts
  - Performance and quality attributes
  - Security, privacy, scalability, sustainability…
- Design trade-offs regarding quality attributes
- Make informed architectural decisions
- Rationale to support architectural decisions
  - Whether to employ a blockchain or some conventional components

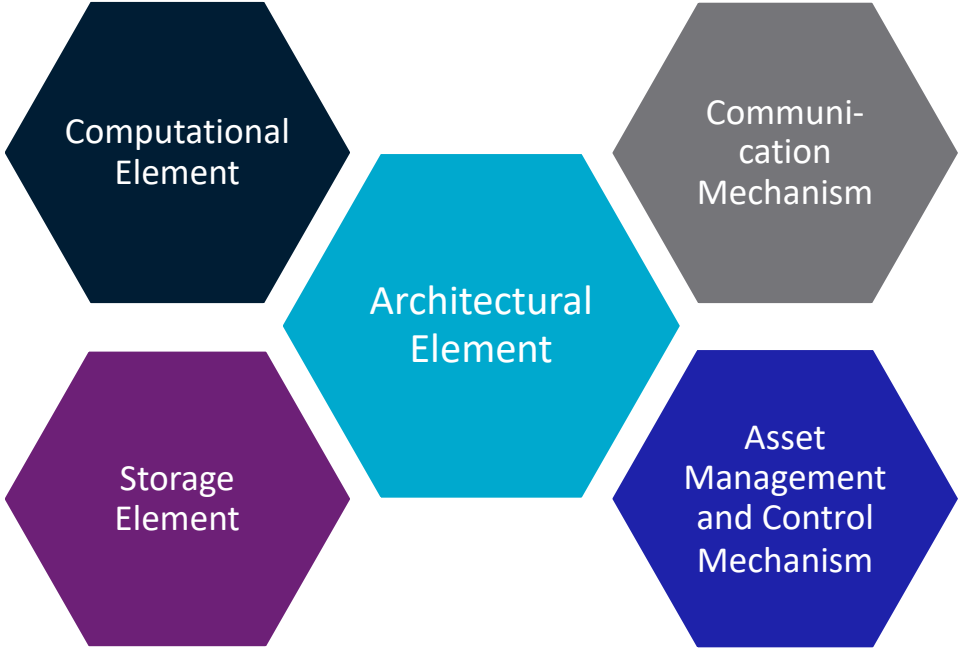
Blockchain as a Software Component

- Complex, network-based software component

- Features
  - Cryptographically-secure payment
  - Mining
  - Transaction Validation
  - Incentive mechanism
  - Permission management

Blockchain's Functions in Application

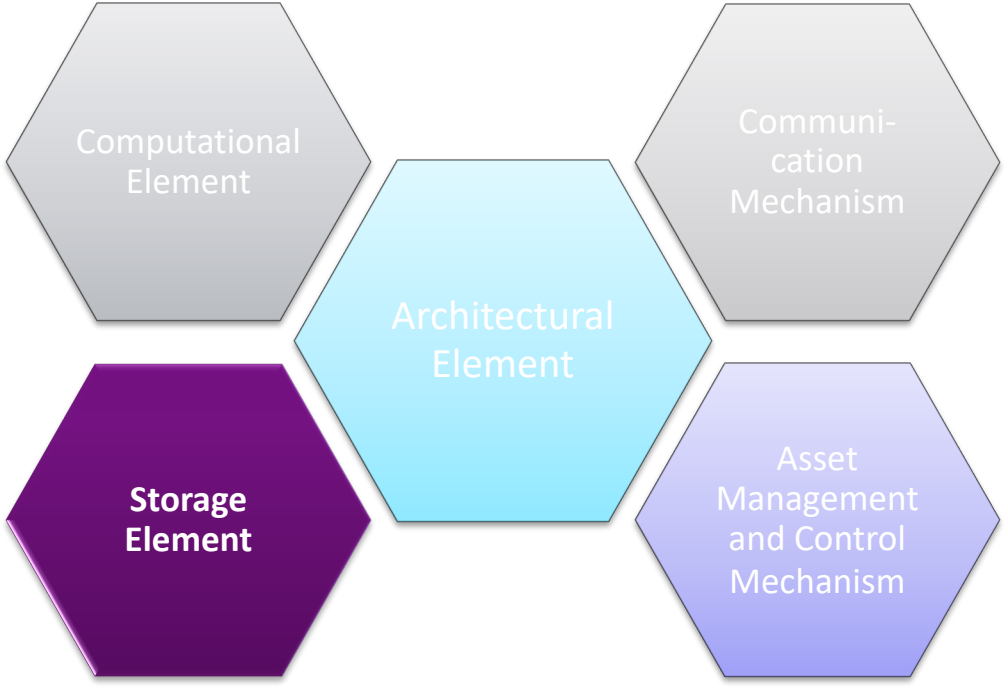
- Blockchain as…



Properties of Blockchain

- Blockchain cannot meet requirements for all usage scenarios
  - E.g. those that require real-time processing
- Fundamental Properties
  - Immutability *from committed transaction*
  - Integrity *from cryptographic tool*
  - Transparency *from public access*
  - Equal rights *from consensus*
- Limitation
  - Data privacy
  - Scalability
    - Size of the data on blockchain
    - Transaction processing rate
    - Latency of data transmission

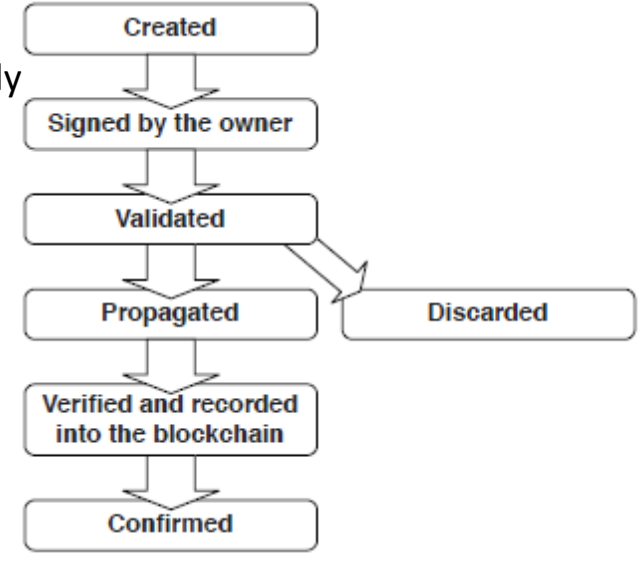
Blockchain's Functions in Application



Blockchain Functionality

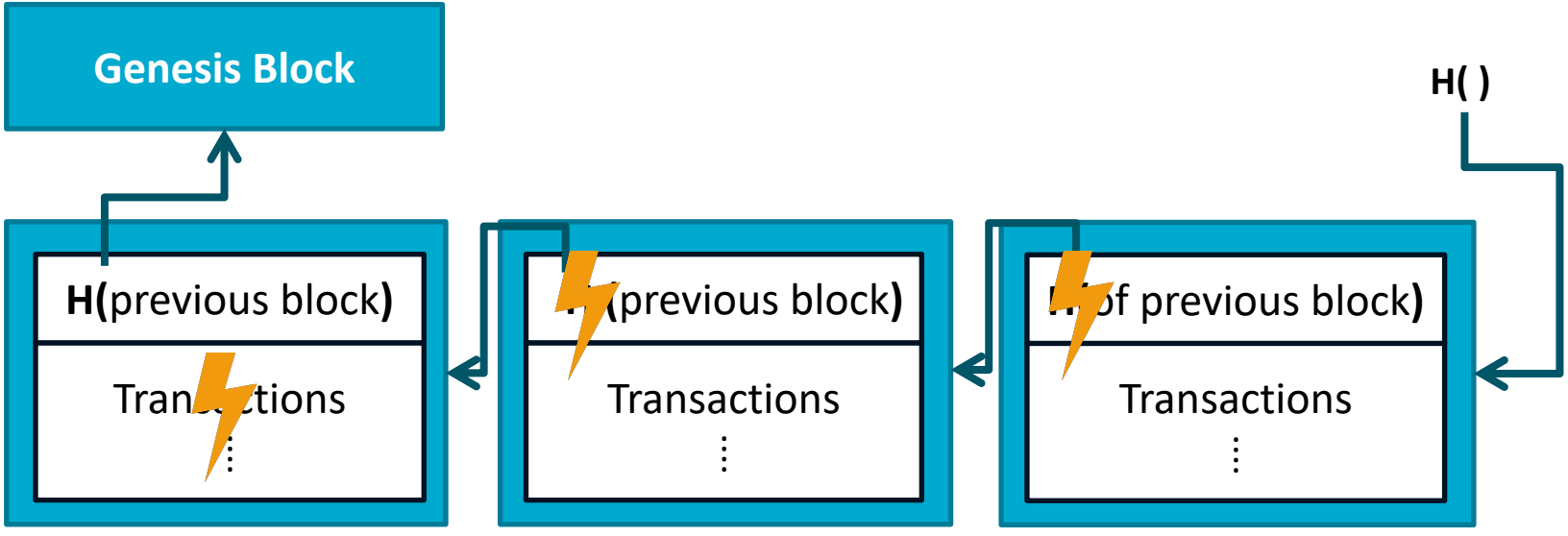
Blockchain as Storage Element

- Append only data store – prevent tampering of data
  - An ordered list of blocks, where each block contains a small (possibly empty) list of transactions.
  - Each block is “chained” back to the previous block, by containing a hash of the previous block
- Transactions
  - Information is recorded within the transactions
  - Update blockchain states
  - Store code, variables, and results of function calls
  - Public key and digital signatures are normally used to identify accounts and to ensure integrity and authorization of transactions



Blockchain as Storage Element

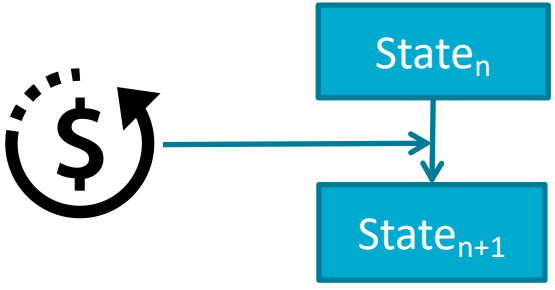
- Blockchain data structure
  - Linked list with hash pointers
- Tamper-proof
  - Computational constraints
  - Incentive scheme



Blockchain as Storage Element

Transactions

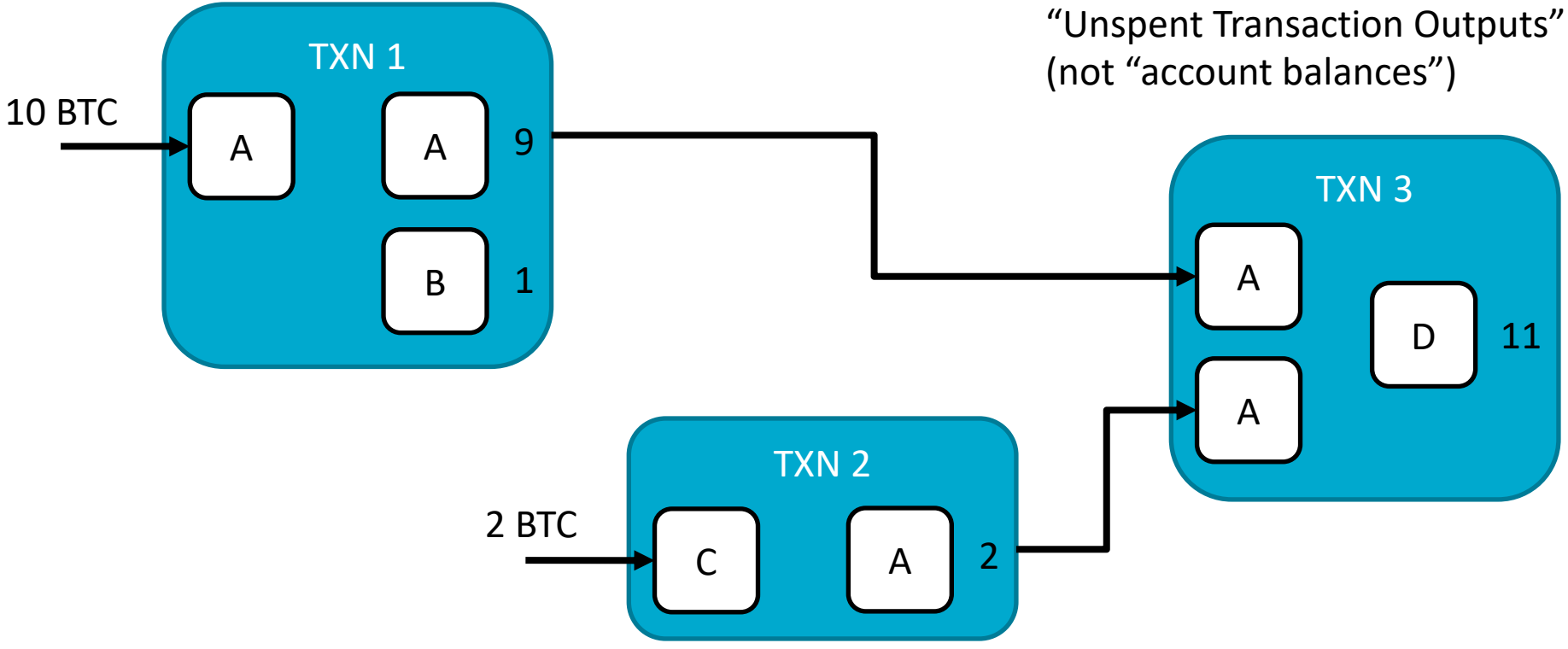
- Transactions represent authorized state transitions
- Transactions can record data
- Transaction can transfer control of digital assets among participants
  - Crypto-currencies
  - Smart contract-based digital assets
- Public key cryptography and digital signature are used to identify accounts
  -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



Representation of Crypto-currency Holdings

- Bitcoin
  - Collection of unspent transaction outputs (UTXO) from all previous transactions to that address
- Ethereum
  - Represented in a global system state
    - Smart contract has its own storage: key-value data store
    - Updates to contract variables do not change the historical data
    - Transactions update the latest view of those variables in the contract store

Bitcoin Holdings – UTXO



What Do Bitcoin Transactions/ Holdings Look Like?

BLOCKCHAIN

Transaction View information about a bitcoin transaction

cb8ecc87d5409bba58939ff55d5644f7354c96a4bd33b184ee17538e7664c66e

17A16QmavnUfCW11DAApiJxp7ARnxN5pGX → 3BMEXG9jyqNgxcA2MDjHWqBJoiPwYJZJPp 0.03049287 BTC

1DLWY19a6VwQGH6SFHuiPHEFHhmRBWDRgw

16dUKAF8VuSrVRJbSDBQypGLTchoKWfnyx 0.0315 BTC

0.04109533 BTC

1E3prxSVxHaxr7kAT1M3ASdqKve3yfY92V

0.06147974 BTC

17A16QmavnUfCW11DAApiJxp7ARnxN5pGX

12.07134555 BTC

5 Confirmations | 12.23591349 BTC

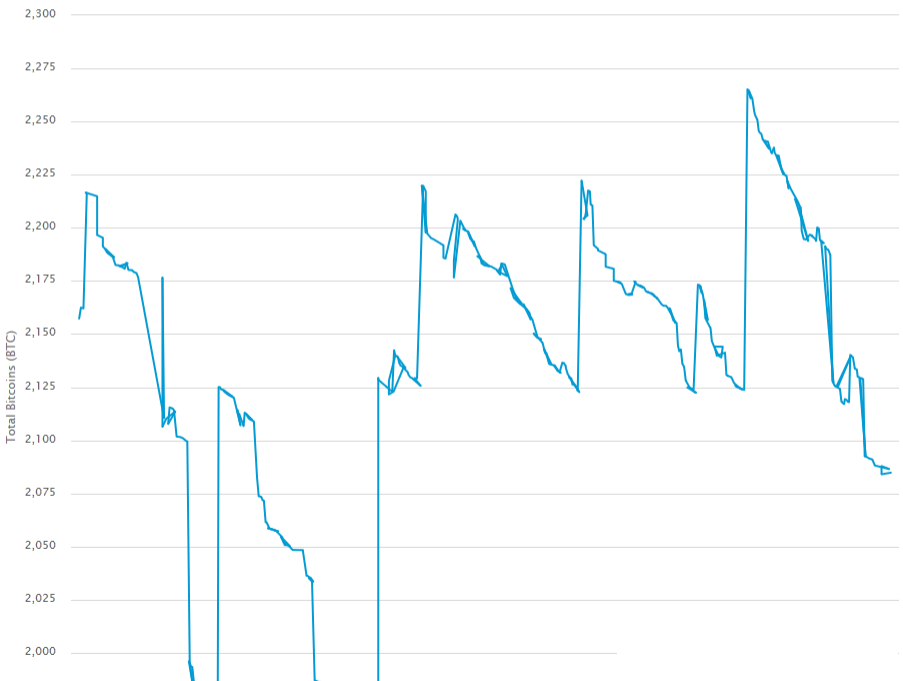
Summary	
Size	358 (bytes)
Weight	1432
Received Time	2018-05-05 12:29:43
Included In Blocks	521319 (2018-05-05 12:40:46 + 11 minutes)
Confirmations	5 Confirmations
Visualize	View Tree Chart

Inputs and Outputs	
Total Input	12.23691349 BTC
Total Output	12.23591349 BTC
Fees	0.001 BTC
Fee per byte	279.33 sat/B
Fee per weight unit	69.832 sat/WU
Estimated BTC Transacted	0.03049287 BTC
Scripts	Show scripts & coinbase

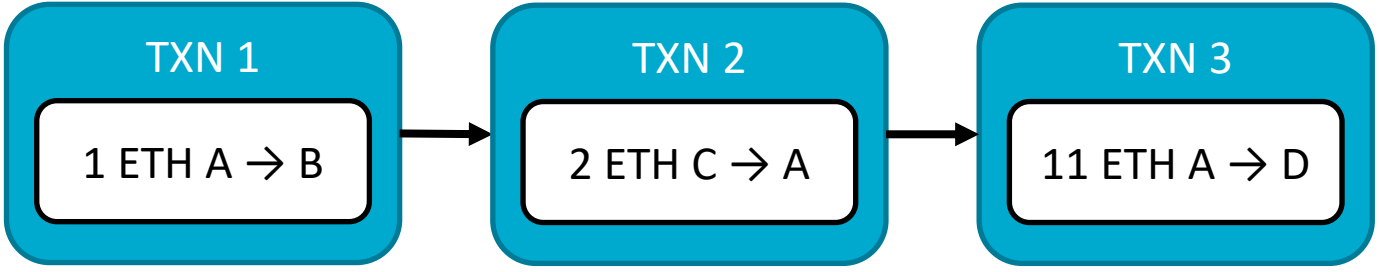
BLOCKCHAIN

Balance of bitcoin address: 17A16QmavnUfCW11DAApiJxp7ARnxN5pGX

Source: blockchain.info



Ethereum Holdings – Accounts



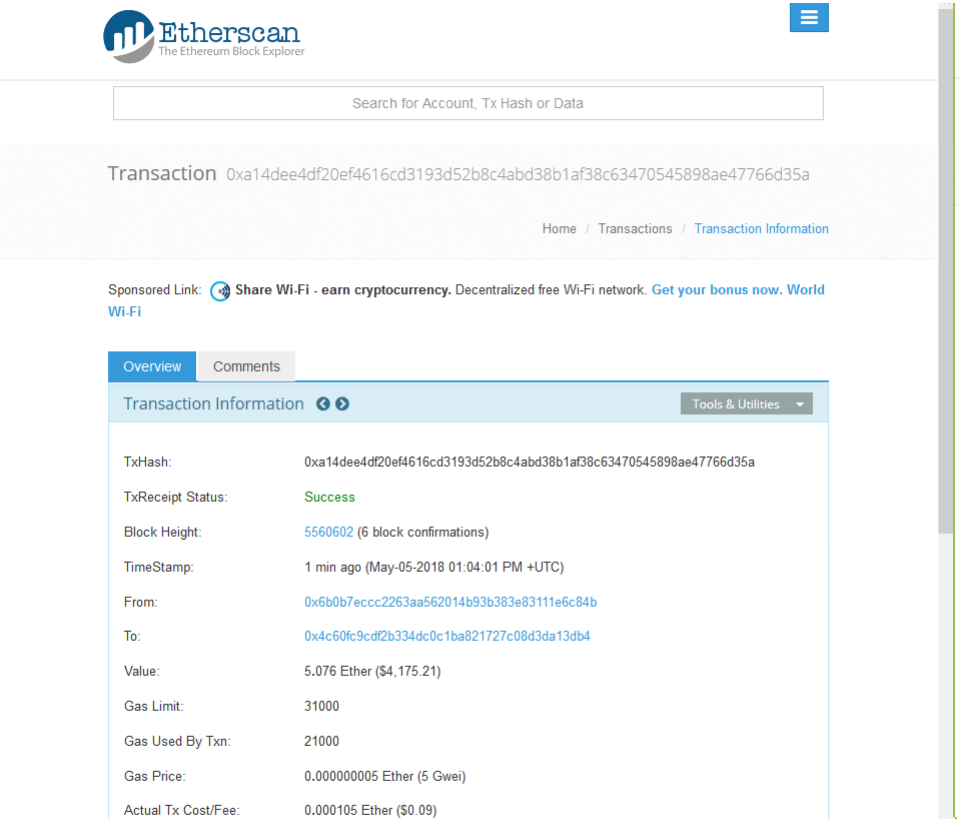
Address	Balance
A	10
B	1
C	4

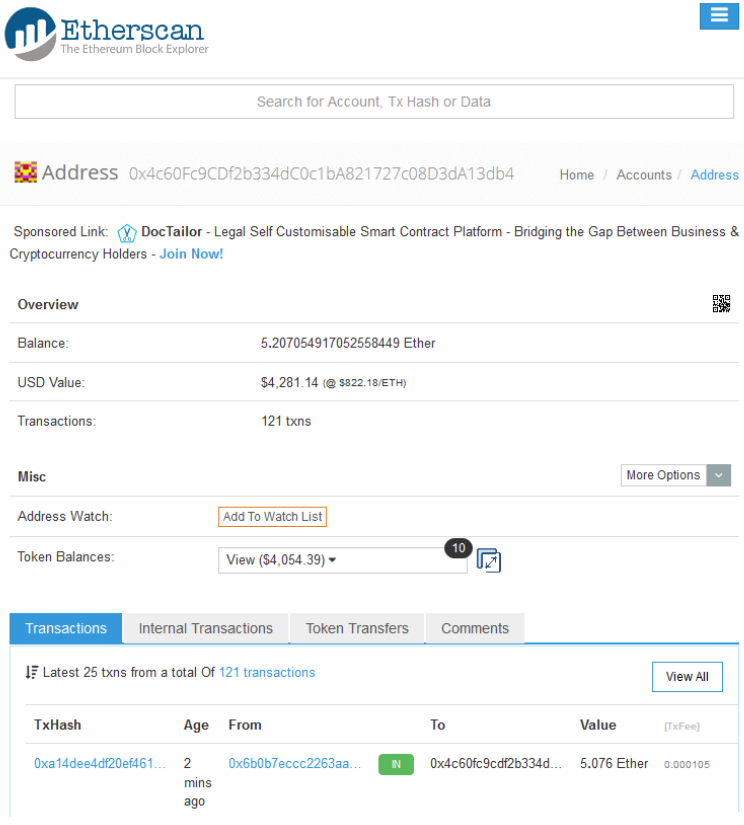
Address	Balance
A	9
B	2
C	4

Address	Balance
A	11
B	2
C	2

Address	Balance
A	0
B	2
C	2
D	11

What Do Ethereum Transactions/ Holdings Look Like?





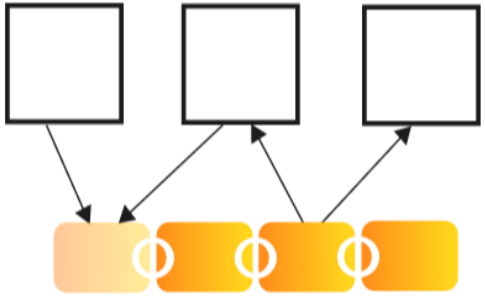
Storing Arbitrary Data on Blockchain

- Two ways of storing data on blockchain
  - Adding data into transaction
    - Bitcoin
    - Ethereum
  - Adding data into contract storage
    - Smart contracts have an address, which is used to invoke the contract
    - Smart contract can only update its own storage
- Both ways store data through submitting transactions
  - Contain information of money transfer
  - Together with optional other data

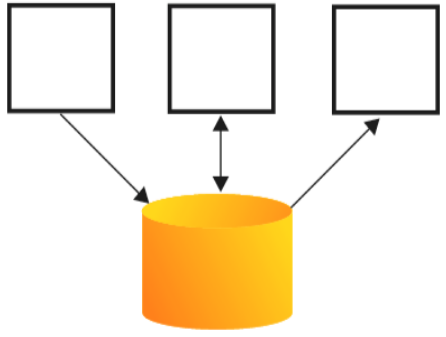
Comparison with Other Data Storage

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

Blockchain vs. Shared DB: *Operations*

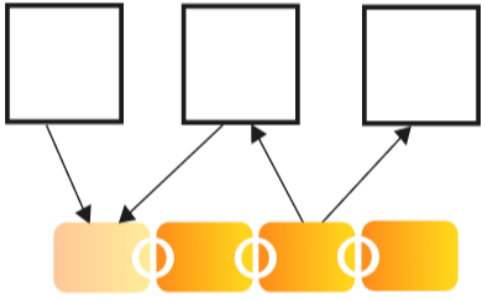


- Insert Transaction (Append Only)
  - Current view of smart contract variable values can mimic database
- Analogy: log
  - Data items get appended
  - Never deleted or updated

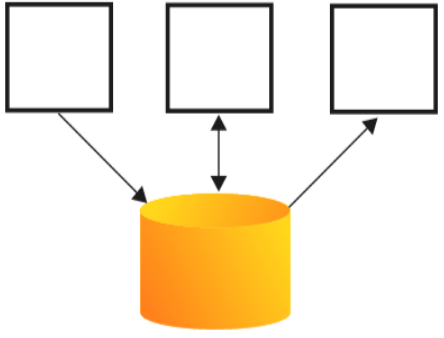


- Create
- Read
- Update
- Delete

Blockchain vs. Shared DB: *Consensus*

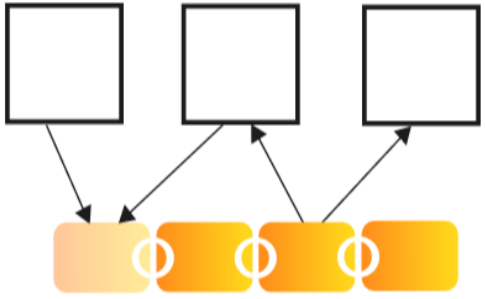


- Majority of peers agree on the outcome of transactions
- Assumption
  - No master
  - No trustworthy node

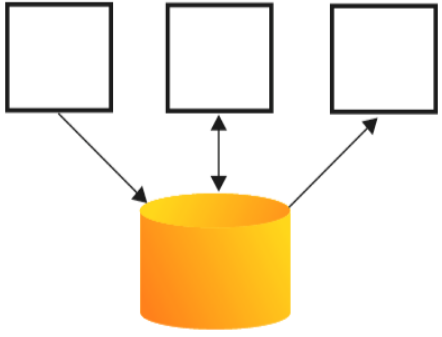


- Distributed Transactions (2 Phase Commit, Paxos)
- Synchronization

Blockchain vs. Shared DB: *Replication*

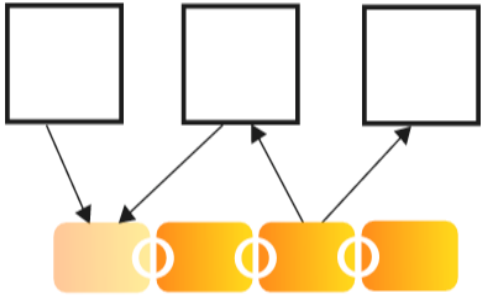


- Full Replication on every peer

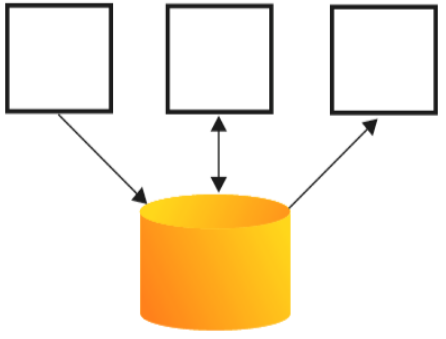


- Master-Slave
- Multi-master

Blockchain vs. Shared DB: *Consistency*



- Transactions validated everywhere
- Global rules enforced on the whole blockchain
  - No extra money created during a spending transaction
  - No money transfer without authorization
- Application-specific rules
  - implemented in smart contract



- Integrity Constraints

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

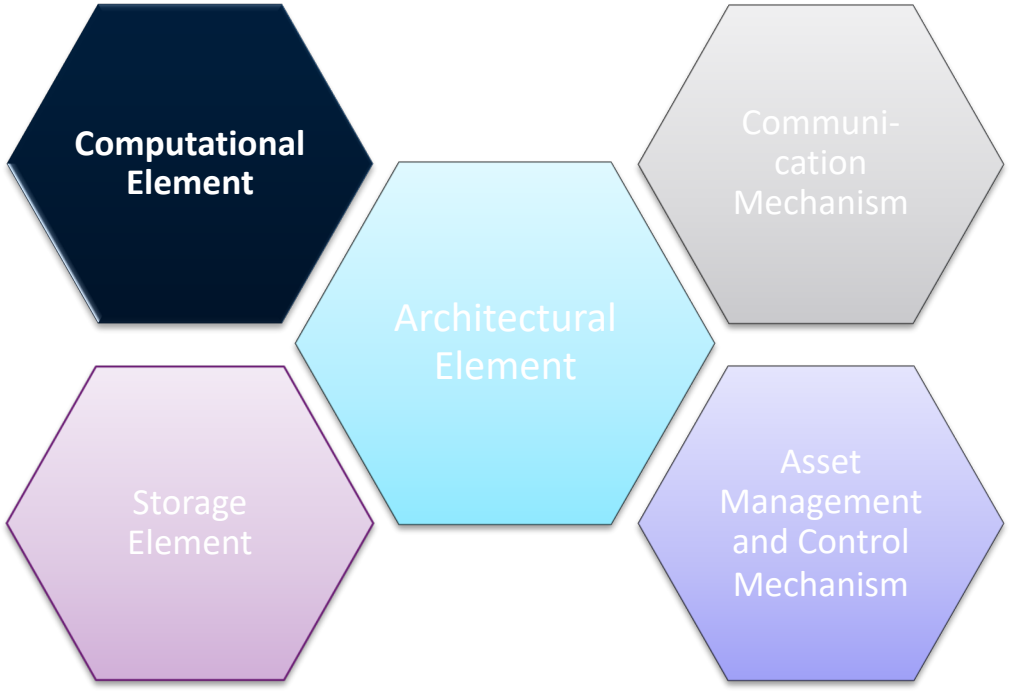
- Peer-to-peer Data Storage allow users to access data that is stored in other computers connected to the same peer-to-peer network
  - BitTorrent, IPFS (InterPlanetary File System)

Blockchain	Peer-to-Peer Data Storage
Very high availability • All nodes have the same shared copy of the blockchain data	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

	Blockchain	Replicated State Machines
Fault Tolerance	Use distribution to not depend on any single entity	• Replicate state at multiple servers • Coordinate service requests from clients
Consensus	Ensure only one among multiple conflicting proposed transaction is included	Decide upon receiving update requests from components • Ensure only one client acquires a lock
Voting	large portion of the community to agree to achieve consensus	A quorum of voters with weighted votes
Replication	Transactions are replicated and persisted after it is included	• Transmitting state update data among components • Information is replicated
Facilitation	Blocks and transactions are totally ordered	Requests are ordered

Blockchain's Functions in Application



Student Task

- Think of a situation when would you:
  - move some computation on-chain
  - keep some computation off-chain

Blockchain Functionality

Blockchain as a Computational Infrastructure

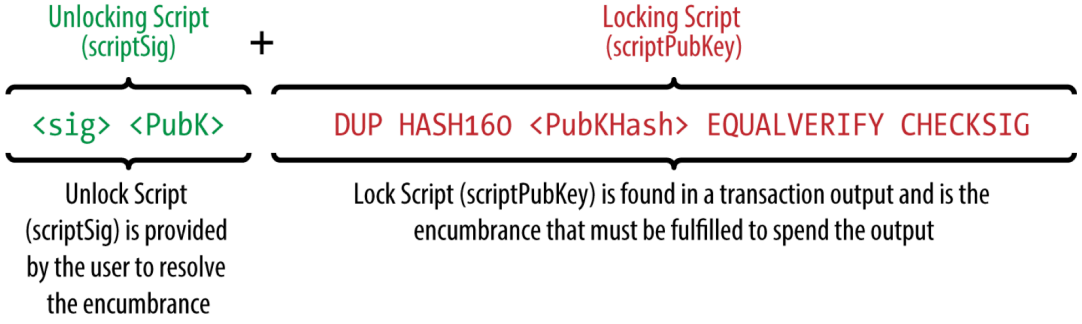
- Architectural decision
  - Which parts of data and computation should **be placed on-chain or kept off-chain**?
    - The amount computation power, data storage space and control of read accesses on a blockchain can be limited
  - A common practice: store hashed data, meta-data and small-sized public data on-chain and keep large and private data off-chain
- Other platforms (e.g. IPFS) can be used for providing a data layer on top of blockchain
- Oracles: interacting with the external world

Data and Computational Platform

- **Blockchain 1.0** – Cryptocurency 
- **Blockchain 2.0** – Cryptoeconomic State Machine HYPERLEDGER ETHEREUM c·rda
  - 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 take custody over assets on the ledger
  - Computational result is stored on the public ledger

Limited Computational Power of 1st Generation

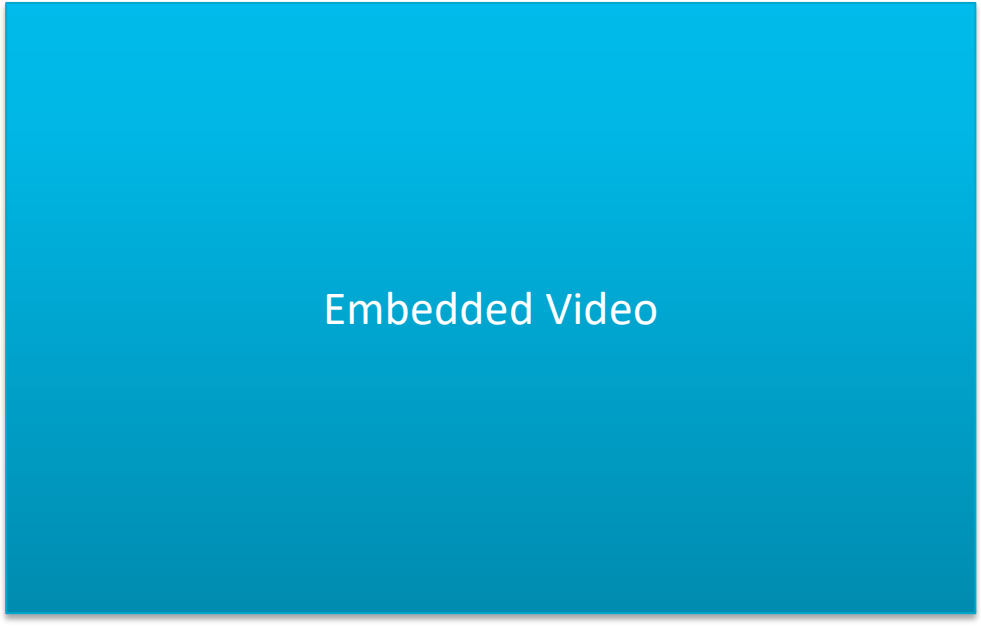
- Native smart contracts on Bitcoin do not support complex control flow definition



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

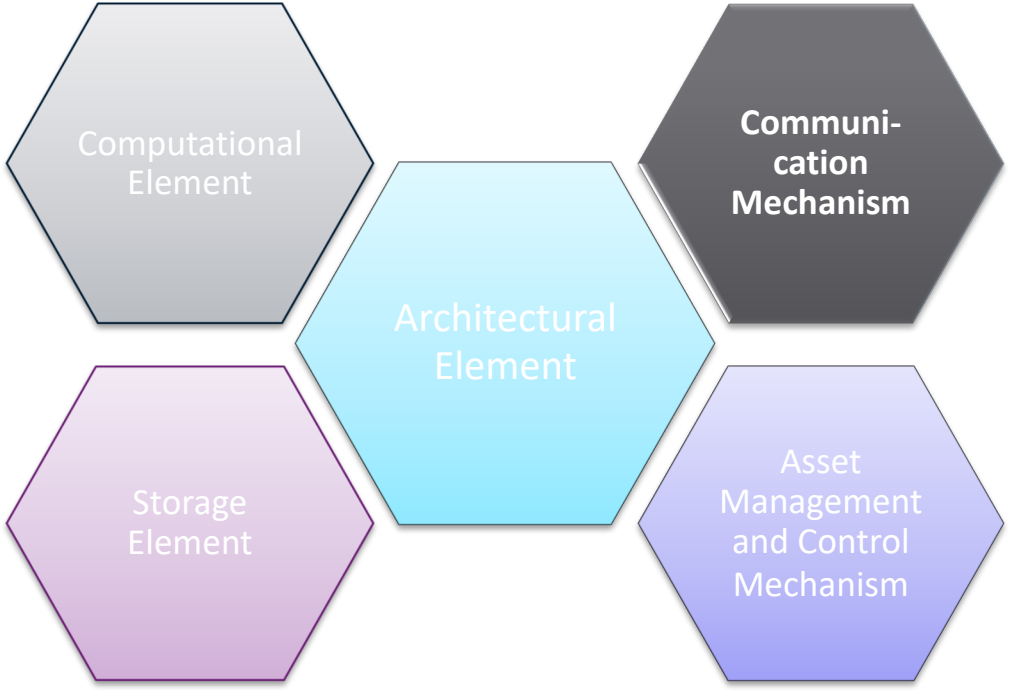
Turing-complete Programming of 2nd Generation

- Ethereum is a general computational platform
  - Turing complete programming language
  - As expressive as every other general purpose programming language
  - Practical limitations on computational complexity
    - Gas limit



Video source: https://www.youtube.com/watch?v=U1XOPIqyP7A

Blockchain's Functions in Application

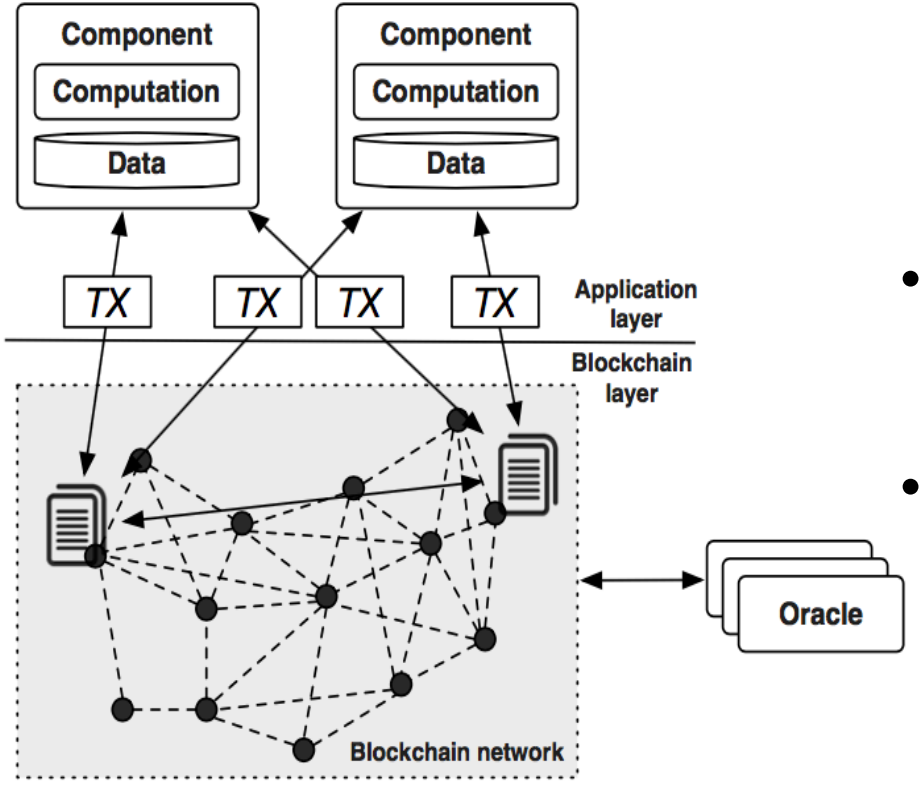


Blockchain Functionality

Blockchain as a Communication Mechanism

- Transactions that are announced get broadcast across the network
- Committing a transaction to the data structure may be required before regarding the transaction as final
  - …or the receiver can already act based on the announcement
  - Threshold for viewing a transaction as committed: user-defined

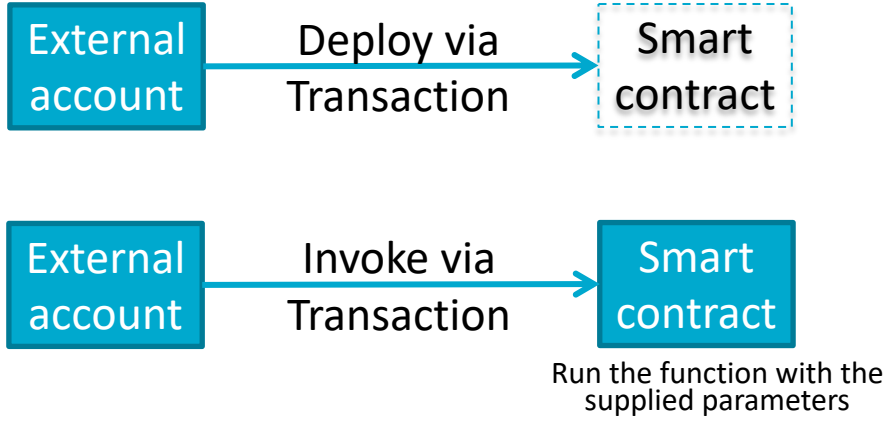
Data Communication



- 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



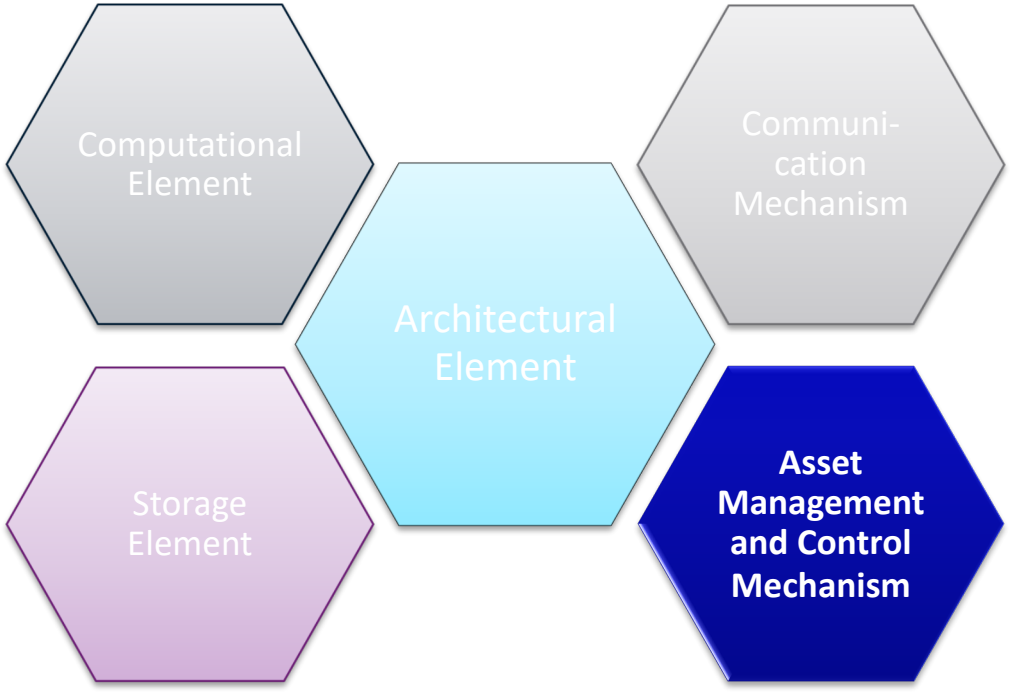
Computation Communication 2/2

- 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

Student Task

- When would you use an Oracle?

Blockchain's Functions in Application



Blockchain Functionality

Blockchain as an Asset Management and Control Mechanism

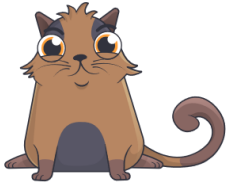
- Cryptocurrency and tokens are virtual assets
- Typically, only the owner can control them
- More complex models can be implemented, e.g.:
  - Multi-signature or threshold voting (more than one account controls an asset)
  - Escrow: smart contract code controls an asset
- Highly customizable through smart contracts

Fungible and Non-fungible Tokens

- Fungible tokens: interchangeable
  - E.g., $2 coin, $10 note
  - Main concern: how many?
  - Ethereum: ERC20 standard
    - https://theethereum.wiki/w/index.php/ERC20_Token_Standard
    - Example: OmiseGO (OMG).

      "The OmiseGO blockchain comprises a decentralized exchange, liquidity provider mechanism, clearinghouse messaging network, and asset-backed blockchain gateway. … It uses the mechanism of a protocol token to create a proof-of-stake blockchain to enable enforcement of market activity amongst participants. Owning OMG tokens buys the right to validate this blockchain, within its consensus rules."

- Non-fungible tokens
  - E.g., houses, cars, patents
  - Main concern: which ones?
  - Ethereum: ERC721
    - https://github.com/ethereum/EIPs/issues/721
    - Example: cryptokitties https://www.cryptokitties.co/
    - Kitties are non-fungible, individual, and their appearance depends on the individual features



Student Task

- Now think about examples of fungible and non-fungible assets that you know of

Asset Management and Control Mechanism

- Tokenization enable asset management
  - Represent digital assets or physical assets

- Tokenization
  - Starts when an asset under custody is represented using a cryptographic token
  - The control of this token aligns with the ownership of the asset
  - The reverse process takes place if the user redeems the token to recover the asset
  - Using smart contract to associate conditions with the transfer of ownership

Asset Management and Control Mechanism

- Native token of the 1st 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 2nd generation of blockchain
  - Enable more expressive data structure
  - Flexibility for tokenizing a wider variety of assets

Summary

- Software Architecture Basics
- Blockchain in Software Architecture

Thank You

- **Xiwei Xu** | Senior Research Scientist
- Architecture & Analytics Platforms (AAP) team
  - **T** +61 2 9490 5664
  - **E** xiwei.xu@data61.csiro.au
  - **W** www.data61.csiro.au/