Architecture for Blockchain Applications – Overview, Design Process, and Design Patterns

Lecture 4 in Ringvorlesung “Blockchain & RPA” @ CAU Kiel

Prof. Dr. Ingo Weber | Chair for Software and Business Engineering

ingo.weber@tu-berlin.de | Twitter: @ingomweber
Book for this lecture: Architecture for Blockchain Applications


Includes definitions of blockchain (concept), DLT, smart contracts, etc.
Blockchain Research with my previous team at Data61 – Overview

**DOMAINS**
- Supply Chain / Food Provenance
- Resource handling (energy, water, land, titles)
- Welfare payments
- International trade

**TECHNOLOGY**
- Designing Trustworthy Blockchain Systems Using Smart Contracts
- Architecting Blockchain applications using Model-driven development of smart contracts
- Distributed Ledger Technologies
- Blockchain platforms
- Smart contracts
- On/Off Blockchain storage

**OUTCOMES**
- Identifying design trade-offs
- Identifying Technical Risks & Opportunities
- Formal and empirical evaluation of Blockchain systems

**GOALS**
- Software Engineering & Architecture
- Business Process Management
- Cyber Security
- Dependability

**PRINCIPLES**
- Identifying design trade-offs
- Identifying Technical Risks & Opportunities
- Formal and empirical evaluation of Blockchain systems
What is the Beef about Blockchain?

Video & reports:
What is a Blockchain?

Visualization of a Blockchain: http://ethviewer.live
Blockchain 2\textsuperscript{nd} gen – Smart Contracts

- 1\textsuperscript{st} gen blockchains: transactions are financial transfers
- Now Blockchain ledger can do that, and store/transact any kind of data
- Blockchain can deploy and execute programs: Smart Contracts
  - User-defined code, deployed on and executed by whole network
  - Can enact decisions on complex business conditions
  - Can hold and transfer assets, managed by the contract itself
  - Ethereum: pay per assembler-level instruction
So what?

• Well, blockchains are exciting because they can be used as a new foundation for re-imagining systems:
  • a neutral infrastructure for processing transactions and executing programs
  • potentially interesting for innovation at all touch-points between organizations or individuals
    ➢ blockchain applications have the potential to disrupt the fabric of society, industry, and government

• Blockchains can also be used as a technology platform to handle hard issues of data replication and system state synchronization with high integrity.
Selected Blockchain Projects

• Australian Securities Exchange:
  • Settlement of trades to be sped up from 2-3 days to minutes, freeing up billions of $$$
  • Under development, industry engagement & testing ongoing
  • Go-live of the blockchain system planned for March or April 2021

• Modum.io:
  • Ensure drugs do not exceed a temperature threshold
    • Tamper-proof IoT device & blockchain storage of data
  • Otherwise: use refrigeration trucks, 4-8x pricier

• Australian National Blockchain
  • Joint initiative by IBM, Data61 CSIRO, HSF, KWM
  • Platform for blockchain-based legal contracts
  • For a fee, implement your business contracts in code
Overview

- Blockchain can be a component of a big software system
- Communicate with other components within the software system
Functions blockchain can provide in an application architecture

• Blockchain as...

Computational Element

Communication Mechanism

Storage Element

Asset Management and Control Mechanism

Architectural Element
Non-Functional Trade-Offs

- Compared to conventional database & script engines, blockchains have:
  - (-) Confidentiality, Privacy
  - (+) Integrity, Non-repudiation
  - (+ read/ - write) Availability
  - (-) Modifiability
  - (-) Throughput / Scalability / Big Data
  - (+ read/ - write) Latency

Security: combination of CIA properties
Model-driven Engineering / BPM

• Business Process Execution & Analysis
  • Runtime verification for business processes utilizing the Bitcoin blockchain, C. Prybila, S. Schulte, C. Hochreiner, I. Weber, FGCS 2018
  • Interpreted execution of business process models on blockchain. O. López-Pintado, M. Dumas, L. García-Bañuelos, I. Weber. EDOC 2019

• Data / Asset Modeling
  • Regerator: a Registry Generator for Blockchain, A. B. Tran, X. Xu, I. Weber, CAISE 2017 Demo

• Combined Asset & Process Modeling
Combining process and data/token models
Tooling: Lorikeet at Design Time
Evaluation of Suitability
Evaluation Framework

- **Is multi-party required?**
  - Yes
  - Is trusted authority required?
    - Yes
    - Is trusted authority decentralizable?
      - Yes
      - Consider Conventional Database
      - No
      - Consider Blockchain
    - No
    - Is operation centralised?
      - Yes
      - Consider Conventional Database
      - No
      - Is immutability required?
        - Yes
        - Can big data be stored off-chain?
          - Yes
          - Consider Blockchain
          - No
          - Is high performance required?
            - Yes
            - Can encrypted data be shared?
              - Yes
              - Consider DLTs
              - No
              - Is transparency required?
                - Yes
                - Consider DLTs
                - No
                - No
              - No
            - No
          - No
        - No
      - No
    - No
  - No
Is Trusted Authority Required? 1/2

• An entity that is relied upon to perform a function (operating a system)
  • Blockchain is **NOT SUITABLE** if a single party can or must be relied upon as a trusted authority
  • Trusted authority implements a traditional centralized solution with conventional technologies

• Scope of the system is important in deciding the question
  • Bank is a trusted authority for bank accounts
  • Central bank is a trusted authority for inter-bank payments

• Trusted authority is a single point of failure
  • Technical single points of failure can be mitigated by using redundancy
  • Single points of organizational or business failure remain present
    • Business failures, service interruptions, data loss or fraud.
• Trusted authority is a monopoly or oligopoly service provider
  • “Rent-seeking” behavior is not uncommon
    • Increase one’s share of existing wealth without creating new wealth
• A natural trusted authority might be difficult for everyone to accept
  → reliance on that party
  • Centralization of services can be perceived as a loss of control or power
• Blockchain is **SUITABLE** for system where no single party is acceptable for operating the system
  • Operated jointly by a collective of nodes
  • Does not remove need for trust entirely
    • Users are exposed to risk in use of blockchain technology
    • What is trusted is the software, the incentive mechanism, and “oracles”
• Distributed Trust
  • Remove the need to trust a single third-party to maintain a ledger.
• *Data immutability* means data cannot be changed or altered after its creation
  • Immutability supports non-repudiation
    • Assurance that a party cannot deny the authenticity of their signature
• Conventional technologies naturally support mutable data
• Blockchain naturally supports data immutability in the ledger
  • Linking of blocks in a chain of cryptographic hashes supports immutability
  • Data continually replicated across many locations and organization
    • Attempts to change it in one location will be interpreted as an attack on integrity
  • Strong evidence that the transactions were performed by someone with control over corresponding cryptographic keys
• Transaction history is immutable, the latest view of the current state can change
Is Immutability Required? 2/3

• Impossible to change the transaction history in most blockchains
  • Causes problems if blockchain contains illegal content
  • Court might order that content be removed from a blockchain

• Other issues
  • Disputed transactions
  • Incorrect addresses
  • Exposure / theft or loss of private keys
  • Data-entry errors
  • Unexpected changes to assets tokenized on blockchain

• Other cheaper mechanisms available to prove the originality of data
  • Hashing technology
  • Cryptographically signed data
  • Time-stamped, distributed / federated data stores
Immutability of Nakamoto-based blockchain is a long-run probabilistic durability property

- Conventional database supports ACID (Atomicity, Consistency, Isolation and Durability)
- A transaction initially thought by a participant to be committed may later turn out to have been on a shorter chain
- A transaction is in practice immutable if it has been committed to a blockchain for a sufficiently long time (number of confirmation blocks)

Blockchain using other consensus mechanism can offer stronger, more conventional immutability

- E.g. Practical Byzantine Fault Tolerance
- Small number of known nodes participating in the operation
Evaluation Framework

Is multi-party required?
- Yes
  - Is trusted authority required?
    - Yes
      - Is trusted authority decentralizable?
        - No
          - Consider Conventional Database
        - Yes
          - Consider Blockchain
    - No
      - Is operation centralised?
        - Yes
          - Consider Conventional Database
        - No
          - Is immutability required?
            - Yes
              - Is high performance required?
                - Yes
                  - Can big data be stored off-chain?
                    - No
                      - Consider DLTs
                    - Yes
                      - Consider Blockchain
                - No
                  - Is transparency required?
                    - Yes
                      - Can encrypted data be shared?
                        - No
                          - Consider DLTs
                        - Yes
                          - Consider Blockchain
                    - No
                      - Consider Conventional Database
Design Process for Blockchain-based Systems
• Every step helps to decide between alternative options
  - A guide through the options of the Taxonomy in Chapter 3 of the book
  - Systematic comparison of different design options
Taxonomy

**Fundamental properties: immutability, non-repudiation, integrity, transparency, equal rights**

Blockchain-related design decisions regarding (de)centralisation, with an indication of their relative impact on quality properties

Legend: ⊕: Less favourable, ⊕⊕: Neutral, ⊕⊕⊕: More favourable

<table>
<thead>
<tr>
<th>Design Decision</th>
<th>Option</th>
<th>Impact</th>
<th>#Failure points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fully Centralised</td>
<td>Services with a single provider (e.g., governments, courts)</td>
<td>⊕</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Services with alternative providers (e.g., banking, online payments,</td>
<td>⊕⊕</td>
<td></td>
</tr>
<tr>
<td></td>
<td>clou services)</td>
<td>⊕⊕</td>
<td></td>
</tr>
<tr>
<td>Partially Centralised &amp;</td>
<td>Permissioned blockchain with permissions for fine-grained operations</td>
<td>⊕⊕</td>
<td></td>
</tr>
<tr>
<td>Partially Decentralised</td>
<td>on the transaction level (e.g., permission to create assets)</td>
<td>⊕⊕</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Permissioned blockchain with permissioned miners (write), but</td>
<td>⊕⊕</td>
<td></td>
</tr>
<tr>
<td></td>
<td>permission-less normal nodes (read)</td>
<td>⊕</td>
<td></td>
</tr>
<tr>
<td>Fully Decentralised</td>
<td>Permission-less blockchain</td>
<td>⊕⊕⊕</td>
<td>Majority</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(nodes, power,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>stake)</td>
</tr>
<tr>
<td>Verifier</td>
<td>Single verifier trusted by the network (external verifier signs valid</td>
<td>⊕⊕</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>transactions; internal verifier uses previously-injected external state)</td>
<td>⊕⊕</td>
<td></td>
</tr>
<tr>
<td></td>
<td>M-of-N verifier trusted by the network</td>
<td>⊕⊕</td>
<td>M</td>
</tr>
<tr>
<td></td>
<td>Ad hoc verifier trusted by the participants involved</td>
<td>⊕</td>
<td>1 (per ad hoc</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>choice)</td>
</tr>
</tbody>
</table>
**Taxonomy**

**Fundamental properties:** immutability, non-repudiation, integrity, transparency, equal rights

Blockchain-related design decisions regarding storage and computation, with an indication of their relative impact on quality properties

<table>
<thead>
<tr>
<th>Design Decision</th>
<th>Option</th>
<th>Fundamental properties</th>
<th>Cost efficiency</th>
<th>Performance</th>
<th>Flexibility</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Item data</strong></td>
<td>On-chain</td>
<td>Embedded in transaction (Bitcoin)</td>
<td>★★★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Embedded in transaction (Public Ethereum)</td>
<td>★★★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Smart contract variable (Public Ethereum)</td>
<td>★★★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Smart contract log event (Public Ethereum)</td>
<td>★★★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td></td>
<td>Off-chain</td>
<td>Private / Third party cloud</td>
<td>★</td>
<td>~KB Negligible</td>
<td>★★★★</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Peer-to-Peer system</td>
<td>★★★</td>
<td>★★★</td>
<td>★★★</td>
</tr>
<tr>
<td><strong>Item collection</strong></td>
<td>On-chain</td>
<td>Smart contract</td>
<td>★★★</td>
<td>★★★ (public)</td>
<td>★★★</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Separate chain</td>
<td>★★★</td>
<td>★ (public)</td>
<td>★★★</td>
</tr>
<tr>
<td><strong>Computation</strong></td>
<td>On-chain</td>
<td>Transaction constraints</td>
<td>★★★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Smart contract</td>
<td>★★★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td></td>
<td>Off-chain</td>
<td>Private / Third party cloud</td>
<td>★</td>
<td>★★★★</td>
<td>★★★★</td>
</tr>
</tbody>
</table>
# Taxonomy

Fundamental properties: immutability, non-repudiation, integrity, transparency, equal rights

## Blockchain-related design decisions regarding blockchain configuration

<table>
<thead>
<tr>
<th>Design Decision</th>
<th>Option</th>
<th>Fundamental properties</th>
<th>Cost efficiency</th>
<th>Performance</th>
<th>Flexibility</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Public blockchain</td>
<td>★★★★</td>
<td>★★★</td>
<td>★★</td>
<td>★★</td>
</tr>
<tr>
<td></td>
<td>Consortium/community blockchain</td>
<td>★★</td>
<td>★★</td>
<td>★★</td>
<td>★★</td>
</tr>
<tr>
<td></td>
<td>Private blockchain</td>
<td>★</td>
<td>★★</td>
<td>★★</td>
<td>★★</td>
</tr>
<tr>
<td></td>
<td>Blockchain</td>
<td>★★</td>
<td>★★★</td>
<td>★★</td>
<td>★★</td>
</tr>
<tr>
<td></td>
<td>GHOST</td>
<td>★</td>
<td>★★★</td>
<td>★★</td>
<td>★★</td>
</tr>
<tr>
<td></td>
<td>BlockDAG</td>
<td>★</td>
<td>★★★</td>
<td>★★</td>
<td>★★</td>
</tr>
<tr>
<td></td>
<td>Segregated witness</td>
<td>★★</td>
<td>★★</td>
<td>★★</td>
<td>★★</td>
</tr>
<tr>
<td></td>
<td>Proof-of-work</td>
<td>★★★★</td>
<td>★</td>
<td>★★</td>
<td>★★</td>
</tr>
<tr>
<td></td>
<td>Proof-of-retrievability</td>
<td>★★★</td>
<td>★</td>
<td>★★</td>
<td>★★</td>
</tr>
<tr>
<td></td>
<td>Proof-of-stake</td>
<td>★</td>
<td>★★★</td>
<td>★★</td>
<td>★★</td>
</tr>
<tr>
<td></td>
<td>BFT (Byzantine Fault Tolerance)</td>
<td>★★</td>
<td>★★★</td>
<td>★★</td>
<td>★★</td>
</tr>
<tr>
<td></td>
<td>Bitcoin-NG</td>
<td>★★</td>
<td>★</td>
<td>★★</td>
<td>★★</td>
</tr>
<tr>
<td></td>
<td>Off-chain transaction protocol</td>
<td>★</td>
<td>★★★</td>
<td>★★</td>
<td>★★</td>
</tr>
<tr>
<td></td>
<td>Mini-blockchain</td>
<td>★</td>
<td>★★</td>
<td>★★</td>
<td>★★</td>
</tr>
<tr>
<td></td>
<td>X-block confirmation</td>
<td>★</td>
<td>★</td>
<td>★★</td>
<td>★★</td>
</tr>
<tr>
<td></td>
<td>Checkpointing</td>
<td>★</td>
<td>★★★</td>
<td>★★</td>
<td>★★</td>
</tr>
<tr>
<td></td>
<td>Original block size and frequency</td>
<td>★★★★</td>
<td>n/a</td>
<td>★★</td>
<td>n/a</td>
</tr>
<tr>
<td></td>
<td>Increase block size / Decrease mining time</td>
<td>★</td>
<td>n/a</td>
<td>★★</td>
<td>n/a</td>
</tr>
<tr>
<td></td>
<td>Merged mining</td>
<td>★★★★</td>
<td>★★</td>
<td>★★</td>
<td>★★</td>
</tr>
<tr>
<td></td>
<td>Hook popular blockchain at transaction level</td>
<td>★</td>
<td>★</td>
<td>★★</td>
<td>★★</td>
</tr>
<tr>
<td></td>
<td>Proof-of-burn</td>
<td>★</td>
<td>★</td>
<td>★★</td>
<td>★★</td>
</tr>
<tr>
<td></td>
<td>Side-chains</td>
<td>★★★★</td>
<td>★</td>
<td>★★</td>
<td>★★</td>
</tr>
<tr>
<td></td>
<td>Multiple private blockchains</td>
<td>★</td>
<td>★★★</td>
<td>★★</td>
<td>★★</td>
</tr>
</tbody>
</table>
## Trade-off Analysis 1/2

<table>
<thead>
<tr>
<th>Quality</th>
<th>Design decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scalability</td>
<td>Blockchain tech platform, block size, block frequency</td>
</tr>
<tr>
<td>Security</td>
<td>Consensus protocol</td>
</tr>
<tr>
<td>Cost efficiency</td>
<td>Public/Consortium/Private blockchain</td>
</tr>
<tr>
<td>Performance</td>
<td>Data structure</td>
</tr>
</tbody>
</table>

- **Design decision impact**
  - A decision may improve the performance of one quality attribute
  - ...but may harm the performance of other quality attributes
Trade-off Analysis 2/2

Encrypting data before storing it on a blockchain
- Increase confidentiality
- Reduce performance, may harm transparency or independent auditability

Storing only a hash of data on-chain and keeping the contents off-chain
- Improve confidentiality and performance
- Partly undermine the benefit of blockchains in providing distributed trust
- May introduce a single point of failure, thus reducing system availability and reliability

Using private blockchain instead of public blockchain
- Allow greater control over the admittance of processing nodes and transaction into the system
- Increase barriers to entry for participation, thus partly reduce some benefit of using blockchain

Higher number of confirmation blocks
- Increase confidence in integrity and durability of transaction
- Harm latency
Decentralization

• Blockchain is used in scenarios...
  • ...where no single trusted authority is required
  • ...where the trusted authorities can be decentralized or partially decentralized

• Spectrum: deployment and operation of a system
  
  - Centralized monopolies
  - Central parties with competition between parties
  - Services provided jointly by consortium
  - Fully open service provision in a peer-to-peer system

• Some components or functions are decentralized while others are centralized
On-chain vs. Off-chain

- Many kinds of data are better stored off-chain
  - Scalability reason—“Big” data
    - “Non-tiny” data may be too large to store on blockchain
  - Confidentiality reason – private data
  - Dealing with legacy database
# Design Decisions and Their Impact

<table>
<thead>
<tr>
<th>Design Decision</th>
<th>Option</th>
<th>Fundamental properties</th>
<th>Impact</th>
<th>Performance</th>
<th>Flexibility</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>On-chain</td>
<td>Embedded in transaction (Bitcoin)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Embedded in transaction (Public Ethereum)</td>
<td>4️⃣ 4️⃣ 4️⃣ 4️⃣</td>
<td>4️⃣ 4️⃣ 4️⃣ 4️⃣</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Smart contract variable (Public Ethereum)</td>
<td>3️⃣ 3️⃣ 3️⃣</td>
<td>3️⃣ 3️⃣ 3️⃣ 3️⃣</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Smart contract log event (Public Ethereum)</td>
<td>3️⃣ 3️⃣ 3️⃣</td>
<td>3️⃣ 3️⃣ 3️⃣ 3️⃣</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Off-chain</td>
<td>Private / Third party cloud</td>
<td>4️⃣</td>
<td>~KB Negligible</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Peer-to-Peer system</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Computation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>On-chain</td>
<td>Transaction constraints</td>
<td>4️⃣ 4️⃣ 4️⃣ 4️⃣</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Smart contract</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Off-chain</td>
<td>Private / Third party cloud</td>
<td>4️⃣</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Computation

- Different levels of expressiveness of on-chain computation
  - Bitcoin only allows simple scripts and conditions
    - To transfer Bitcoin payment, script must result in “true”
  - Ethereum provides a Turing complete programming language
    - Scope extends far beyond conditional payments
    - Make modification to the working data in smart contract variables
  - DAML (Digital Asset Modelling Language)
    - More expressive than Bitcoin Script
      - Purposefully not Turing-complete: codify financial rights and obligations
        - Allows static analysis

- Benefit of on-chain computation
  - Inherent interoperability among the systems built on the same blockchain network
  - Neutrality of execution environment
  - Immutability of the program code once deployed

Non-determinism (in practice)
- Code is deterministic, but:
  - Blocks impose an order on transactions, by default not controlled
  - Resulting non-determinism might affect the execution results
Blockchain Selection

- For blockchain platform selection, consider:
  - Requirements of the use case
  - Characteristics of blockchain platform
  - Trade-off analysis

- Consensus protocol and other decisions are typically fixed once blockchain is selected
  - Hyperledger Fabric is an exception
    - Supports pluggable implementations of various consensus protocols
  - Inter-block time is configurable for many consensus protocols
    - For Proof-of-Work: through adjustments to the difficulty of mining, with difficulty adaptation disabled
**Deployment**

- Deployment has impact on quality attributes
  - Deploying on cloud or using BaaS (Blockchain-as-a-Service)
    - Introduces uncertainty of cloud infrastructure
    - If all nodes use the same cloud provider, it becomes a trusted third-party and a single point of failure
  - Deploying a public blockchain on a VPN (virtual private network)
    - Becomes a private blockchain
    - Permissioned access controls provided at the network level
    - VPN introduces additional network latency overhead

---

[Diagram of deployment process]

- **Evaluation of Suitability**
  - Has trusted authority?
    - Yes: How to decentralize the authority?
      - Storage and computation: on-chain vs. off-chain
        - Which blockchain?
          - Block configuration
            - Deployment and Operation
    - No: Decentralization
      - Mutually/immutable data
      - Big/small data
      - Non-critical/critical data
      - Raw/Encrypted data
Operation

• Operation challenges
  - Blockchain-based systems can be harder to modify than conventional systems
  - Blockchain platform software running on multiple independently-operating nodes
  - Updating software can be physically and administratively difficult to coordinate
  - Blockchain is immutable by design, so cannot be retrospectively updated
    • Trust is derived partly from the fact that the smart contract cannot be changed easily

• Governance
  - Changes may be made to correct defects, add features or migrate to new IT context
  - More like diplomacy in multi-party system with no single owner
  - Blockchain is NOT SUITABLE for a system that needs to change frequently
  - Learn from governance in open-source software
  - Governance concerns software development, deployment and operation
  - Immutable smart contract also simplify governance to some degree
    • Smart contract is available for execution while the blockchain operates normally
Summary

• Suitability needs to be evaluated first
  - Fundamental properties and limitations of blockchain
• Each step of the design process: design decision(s)
  - On-chain vs. off-chain
  - Subsidiary design decisions
Design Patterns for Blockchain Applications
What is Design Pattern?

• To solve a recurring problem in software development
• Not a finished design that can be transformed directly into code
• A description or template for how to solve a problem
  • Define constraints that restrict the roles of architectural elements
    • Processing
    • Connectors
    • Data
  • Define constraints that restrict the interaction among these elements
• Causes trade-offs among quality attributes
Advantages of Patterns

• Design patterns can speed up the development process by providing tested, proven development paradigms.
  • Design patterns document the efforts of the experts
  • Design patterns concerns with a flexible software architecture

• Effective software design requires considering issues that may not become visible until later in the implementation.

• Reusing design patterns helps to prevent subtle issues that can cause major problems and improves code readability for programmers and architects familiar with the patterns.
Pattern Collection (Chapter 7)

**Interaction with External World**
- Centralized Oracle
- Decentralized Oracle
- Voting
- Legal and Smart Contract Pair
- Reverse Oracle

**Data Management**
- Off-chain Data Storage
- State Channel
- Encrypting On-chain Data
- Tokenisation

**Security**
- Dynamic Authorization
- Multiple Authorization
- X-confirmation
- Security Deposit

**Contract Structural Patterns**
- Contract Registry
- Embedded Permission
- Data Contract
- Factory Contract
- Incentive Execution

**Deployment**
- Semi-dapp
- dapp

Mixed on and off-chain

Purely on-chain
Pattern 1: Centralized Oracle

• **Solution**
  - Oracle assists in evaluating conditions that cannot be expressed in a smart contract
  - Oracle injects the result to the blockchain in a transaction signed using its own key pair
  - Validation of transactions is based on the authentication of the oracle
Pattern 2: Decentralized Oracle

- **Solution**
  - Decentralized oracle based on multiple servers and multiple data sources
  - Consensus on the external status, e.g.:
    - K-out-of-M threshold signature
Pattern 4: Reverse Oracle

• Solution
  • Transaction ID and Block ID can be integrated into existing system
  • Validation is on blockchain using smart contract
  • An off-chain component is required to query blockchain, which serves as store of current variable values
    • e.g., who is allowed physical access to a device
Pattern 5: Legal and Smart Contract Pair

• Solution
  • SC implements conditions defined in the legal agreement
    • Checked and enforced by the smart contract
  • SC has a blank variable to store hash of legal contract
  • SC address included in the legal agreement
  • Legal agreement hash is added to the SC variable, which is immutable after first write
Thank you for your attention!

Architecture for Blockchain Applications – Overview, Design Process, and Design Patterns

Lecture 4 in Ringvorlesung “Blockchain & RPA” @ CAU Kiel

Prof. Dr. Ingo Weber | Chair for Software and Business Engineering

ingo.weber@tu-berlin.de | Twitter: @ingomweber