Blockchain Application Design and Development, and the Case of Programmable Money

CLOSER’21 Keynote

Prof. Dr. Ingo Weber | April 2021

ingo.weber@tu-berlin.de | linkedin.com/in/ingomweber/ | Twitter: @ingomweber
Agenda

• Blockchain basics and terminology
• Designing and developing blockchain applications
  • Architecture & design
  • Model-driven engineering
• Blockchain and Services:
  • Integrating Blockchain-based Applications with Services
  • Blockchain-as-a-Service
  • Service-orientation vs. Smart Contracts
• Programmable money
• Blockchain adoption
Preliminaries and Definitions
Blockchain – replacing centralized trusted authority

Centralized Trusted Authority

Traditional trusted environment

Blockchain network

Blockchain trustless environment
Blockchain 2nd gen – Smart Contracts

- 1st 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

Approved
So what?

• Well, blockchains are exciting because they can be used as a new foundation for re-imagining systems:
  • 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.
What is a blockchain?

Parable of the blind men and the elephant, see e.g., https://wildequus.org/2014/05/07/sufi-story-blind-men-elephant/ (source of figure)
What is a blockchain?

Parable of the blind men and the elephant, see e.g., [https://wildequus.org/2014/05/07/sufi-story-blind-men-elephant/](https://wildequus.org/2014/05/07/sufi-story-blind-men-elephant/) (source of figure)
Defining Blockchain (1)

• **Distributed Ledger**
  - An “append-only” transaction store distributed across machines (*immutability*)
  - A new transaction might reverse a previous transaction, but both remain part of the ledger

• **Blockchain**
  - A distributed ledger structured into a linked list of blocks
  - Each block contains an ordered set of transactions
  - Use cryptographic hashes to secure the link from a block to its predecessor
Defining Blockchain (2)

• A **Blockchain System** consists of
  • A blockchain network of nodes
  • A blockchain data structure
    • For the ledger replicated across the blockchain network
    • Full nodes hold a full replica of the ledger
  • A network protocol
    • Defines rights, responsibilities, and means of communication, verification, validation, and consensus across the nodes in the blockchain network
    • Includes ensuring authorisation and authentication of new transactions, mechanisms for appending new blocks, incentive mechanisms
Defining Blockchain (3)

• A **Public Blockchain** is a blockchain system with the following characteristics:
  • Has an open network
    • Nodes can join and leave without requiring permission from anyone
  • All full nodes can verify new transactions and blocks
  • Incentive mechanism to ensure the correct operation
    • Valid transactions are processed and included in the ledger and invalid transactions are rejected

• A **Blockchain Platform** is the technology needed to operate a blockchain
  • Blockchain client software for processing nodes
  • The local data store
  • Alternative clients to access the blockchain network
Decentralised Applications and Smart Contracts

**Smart contracts**
- Programs deployed as data and executed in transactions on the blockchain
- Blockchain can be a computational platform (more than a simple distributed database)
- Code is deterministic and immutable once deployed
- Can invoke other smart contracts
- Can hold and transfer digital assets

**Decentralized applications or dapps**
- Main functionality is implemented through smart contracts
- Backend is executed in a decentralized environment
- Frontend can be hosted as a web site on a centralized server
  - Interact with its backend through an API
- Could use decentralized data storage such as IPFS
- “State of the dapps” is a directory recorded on blockchain: [https://www.stateofthedapps.com/](https://www.stateofthedapps.com/)
• **Definition 1 (Distributed Ledger).** A Distributed Ledger is an *append-only store of transactions* which is distributed across many machines.

• **Definition 2 (Blockchain (Concept)).** A Blockchain is a *distributed ledger* that is structured into a *linked list of blocks*. Each block contains an ordered set of transactions. Typical solutions use cryptographic hashes to secure the link from a block to its predecessor.
• **Definition 3 (Blockchain System).** A Blockchain System consists of:
  • a *blockchain network* of machines, also called *nodes*;
  • a *blockchain data structure*, for the ledger that is replicated across the blockchain network. Nodes that hold a full replica of this ledger are referred to as *full nodes*;
  • a network *protocol* that defines rights, responsibilities, and means of communication, verification, validation, and consensus across the nodes in the network. This includes ensuring *authorization and authentication* of new transactions, mechanisms for appending new blocks, incentive mechanisms (if needed), and similar aspects.
• **Definition 4 (Public Blockchain).** A Public Blockchain is a *blockchain system* that has the following characteristics:

  • it has an *open network* where nodes can join and leave as they please without requiring permission from anyone;
  
  • all full nodes in the network can *verify each new piece of data* added to the data structure, including blocks, transactions, and effects of transactions; and
  
  • its protocol includes an *incentive mechanism* that aims to ensure the correct operation of the blockchain system including that valid transactions are processed and included in the ledger, and that invalid transactions are rejected.
• **Definition 5 (Blockchain Platform).** A blockchain platform is the *technology needed to operate a blockchain*. This comprises the blockchain client software for processing nodes, the local data store for nodes, and any alternative clients to access the blockchain network.

• **Definition 6 (Smart Contract).** Smart contracts are *programs* deployed as data in the blockchain ledger, and executed in transactions on the blockchain. Smart contracts can *hold and transfer digital assets* managed by the blockchain, and can invoke other smart contracts stored on the blockchain. Smart contract code is *deterministic and immutable* once deployed.

• **Definition 7 (dapp).** A decentralized application or dapp is a software system that is designed to provide its main functionality through smart contracts.
Includes the definitions from the previous slides.
Cryptocurrencies and Tokens

• Cryptocurrencies
  • ‘Baked in’ to the core platform of public blockchains - base currency of blockchains
  • Symbiotic relationship
    • Blockchain keeps track of the ownership of portions of that currency, e.g. Alice owned 2 Ether, transferred 1 Ether to Bob, offered 0.01 Ether to miner
    • Cryptocurrency enables the incentive mechanism for blockchain operations

• Digital tokens
  • Created and exchanged using smart contracts
  • Represent assets
    • Fungible asset: individual units are interchangeable, e.g. company share, gold
    • Non-fungible asset: represents a unique asset, e.g. cryptokitties, car title

• Not all applications are the same:
  • Transferring coins / tokens vs. tracking movement of physical goods
  • Core difference: where is the default version of the truth, on or off-chain?
Designing and Developing Blockchain Applications
Many interesting applications for Blockchain

- Basically of interest in most lack-of-trust settings where a distributed application can coordinate multiple parties
- Examples:
  - Supply chains
  - Handling of titles, e.g., land, water, vehicles
  - Identity
- Many startups and initiatives from enterprises / governments

... but also many challenges

- When to use blockchain
- Trade-offs in architecture
  - Downsides: cost, latency, confidentiality
  - What to handle on-chain, what off-chain?
Work with my former and my new teams

• Architecting applications on Blockchain:
  • Book [1]
  • Taxonomy and design process [5]
  • “Cost of Distrust”: how much more expensive is blockchain? [7]
    • On some blockchains, cost and throughput are tightly linked
  • Availability analysis from viewpoint of dapps [11]
  • Latency: simulation under changes [12]
  • Multi-tenant applications on blockchain [13]

• Model-driven development of smart contracts
  • Business process execution (including the tools Caterpillar [6] and Lorikeet [10])
  • Model-based generation of code for data structures, non-fungible and fungible tokens, and UI components
  • Data extraction and analytics, e.g. Process Mining on blockchain data [3,4]

• Blockchain Patterns – reusable experience & inspiration [14,15]
  • [URL](https://research.csiro.au/blockchainpatterns/)
  • ...


Functions blockchain can provide in an application architecture

• Blockchain as...

- Computational Element
- Storage Element
- Communication Mechanism
- Asset Management and Control Mechanism
- Architectural Element
Blockchains are Not Stand-Alone Systems

UI for humans

Key management

IoT integration

Blockchain is a component

Legacy systems

BIG DATA

private data

Auxiliary databases
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
Design process

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

- **Is multi-party required?**
  - Yes → **Is trusted authority required?**
    - Yes → **Is trusted authority decentralizable?**
      - No → **Use Conventional Database**
      - Yes → **Use Blockchain**
  - No → **Use Conventional Database**

- **Is operation centralised?**
  - No → **Use Conventional Database**
  - Yes → **Is immutability required?**
    - Yes → **Is high performance required?**
      - Yes → **Is transparency required?**
        - No → **Use DLTs**
        - Yes → **Use Blockchain**
    - No → **Use Blockchain**

- **Can big data be stored off-chain?**
  - Yes → **Use Blockchain**
  - No → **Use DLTs**

- **Can encrypted data be shared?**
  - Yes → **Use Blockchain**
  - No → **Use DLTs**
Designing and Developing Blockchain Applications: Model-driven Engineering for Blockchain Applications
Model-driven Engineering & Blockchain

• Model-driven engineering (MDE):
  • A methodology for using models at various levels of abstraction and for different purposes during software development
  • Low-level models: production code can be directly derived from the models
  • High-level models: means of communication between business owners and developers implementing a system
  • Intermediate levels can support model-based system analysis or system management tools
  • Any level: generate a code skeleton or early version of the code
  • Can cover static structures (like data models) or dynamic behavior (activity sequences)

• Advantages in the blockchain context:
  • Code generation can implement best practices and well-tested building blocks
    • Code can adhere to blockchain “standards” (like ERC-20, ERC-721, …)
  • Models can be independent of specific blockchain technologies or platforms
  • Models are often easier to understand than code – particularly useful in communicating with business partners about smart contracts
    • Facilitates building trust
MDE for data structures and tokens

• Approach:
  • Model data structure (variables, types) – not for fungible tokens
  • Model relationships to other types / tokens
  • Select features
  → Code is generated – deploy or customize

• Feature examples:
  • Fungible tokens:
    • Can be minted? Burnt? By whom?
  • Non-fungible tokens
    • Include standard method(s) for sale
    • One contract for all tokens or one per token?

• Code generated is compliant with standards
  → interface syntax and semantics
MDE for Processes – Motivation

• Integration of business processes across organizations: a key driver of productivity gains

• Collaborative process execution
  • Doable when there is trust – supply chains can be tightly integrated
  • Problematic when involved organizations have a lack of trust in each other
    → if 3+ parties should collaborate, where to execute the process that ties them together?
  • Common situation in “coopetition”
Motivation: example

Issues:
- Knowing the status, tracking correct execution
- Handling payments
- Resolving conflicts
Motivation: example

Issues:
- Knowing the status, tracking correct execution
- Handling payments
- Resolving conflicts
**Approach in a nutshell [2]**

- **Goal:** implement collaborative business processes as smart contracts
  - Translate (enriched) BPMN to smart contract code
  - Triggers act as bridge between Enterprise world and blockchain
  - Smart contract provides:
    - Independent, global process monitoring
    - Conformance checking and process enforcement: only expected messages are accepted, only from the respective role
    - Automatic payments & escrow
    - Data transformation
    - Encryption

- **Processes can make use of data / token contracts**
  - Process activity to hand over title to a car / shipment / grain / ..., e.g., in exchange for fungible tokens
Combining process and data/token models
Data61 tool: Lorikeet [10]

- Lorikeet: automatic generate smart contracts from BPMN models/registry data schema
Design time for fungible tokens
Design time for data models / non-fungible tokens
Design time for process models

Demo video: https://drive.google.com/file/d/1rpy-oHbDVkXa6u4Fn73wSX8rINn1sv3U/view
Runtime View of Process Instances
General remarks about developing blockchain applications

• Code is immutable!

• Consequences:
  • Follow all security best practices
  • Test heavily
  • Do code reviews
  • Build in features for updating as needed and acceptable for the user base
    • Governance for updates, e.g.: updates will become active only after 1 week / 1 month, ...
  • Understand all (relevant) parts of the blockchain system – if you get it wrong, there is no safety net
  • Design includes potentially hard trade-offs between confidentiality and transparency, though patterns exist for resolving parts of those
Integrating Blockchain-based Applications with Services
Blockchain is a closed-world system

• To interact with smart contracts on blockchain, need to:
  • Write: create and broadcast a blockchain transaction (BCTX) for each method call
  • Read: monitor smart contract variable values and/or event logs to see updates

• The outside world speaks Services
  • REST / SOAP-WSDL / JSON RPC

• How to bridge between the two worlds?
  • Recurring problem
  • Our solution: a Trigger component as bridge
Trigger as bridge between blockchain and services
Decentralization

Blockchain

Smart contract
- Code
- Variables
- Event log

Org 1
- App
- Trigger

Org 2
- App
- Trigger
Blockchain-as-a-Service
• Blockchain is a relatively new technology with steep learning curve
  • Gartner survey: “23 percent of [relevant surveyed] CIOs said that blockchain requires the most new skills to implement of any technology area, while 18 percent said that blockchain skills are the most difficult to find.”
• aaS offers can bootstrap that learning phase to a degree
  • Pre-made templates
  • Management tools
    • IDEs
    • Monitoring tools
    • ...
Commercial Offers

• But: what if all nodes are using the same provider?
  • Decentralization?
Unified approach: uBaaS [8]

• Deployment as a service
  • Includes a blockchain deployment service and a smart contract deployment service
  • Platform agnostic to avoid lock-in to specific cloud platforms

• Design patterns as a service
  • Common data management services and smart contract design services
  • Based on a design pattern to better leverage the unique properties of blockchain (i.e. immutability and data integrity, transparency) and address the limitations (i.e. privacy and scalability)
Service-orientation vs. Smart Contracts
Microservice Architecture

- Each user request is satisfied by some sequence of services
- Most services are not externally available
- Each service communicates with other services through service interfaces
- Service depth may be 70, e.g., LinkedIn
Smart Contracts as Services?

• Analogy:
  • Smart contract code ≈ Java Class
  • Deployed smart contract ≈ Java Object, but with some properties
    • Defined interface
    • Standard way to invoke
    • Callable by anyone (who can send transactions to the blockchain)
      → Similar to Web service!

• Some design principles can apply
Service-Oriented Design Principles

- **Standardized Service Contract**: the public interfaces of a service must make use of contract design standards. (Contract: WSDL in WS*)

- **Service Loose Coupling**: to impose low burdens on service consumers (coupling ∼ degree of dependency)

- **Service Abstraction**: “to hide as much of the underlying details of a service as possible”

- **Service Reusability**: services contain agnostic logic and “can be positioned as reusable enterprise resources”

- **Service Autonomy**: to provide reliable and consistent results, a service must have strong control over its underlying environment

- **Service Statelessness**: services should be “designed to remain stateful only when required.”

- **Service Discoverability**: “services are supplemented with communicative meta data by which they can be effectively discovered and interpreted.”

- **Service Composability**: “services are effective composition participants, regardless of the size and complexity of the composition.”

- **Fundamental requirement – interoperability of services**: “...stating that services must be interoperable is just about as evident as stating that services must exist.”

Microservice Principles

✓ Small, focused functionality
✓ Split of responsibility
~ Full-stack & independently updatable without downtime
× Stateless

• While some design principles for Microservice Architectures apply, others do not
  • Updates can be independent
  • But reliance on the inability of anyone to update without agreement / governance is one source of trust in a smart contract
Selected Applications & Adoption
Selected Blockchain Projects

- **Australian Securities Exchange:**
  - Settlement of trades to be sped up from 2-3 days to minutes, freeing up billions of $$
  - In industry engagement, revision based on feedback and testing ongoing
  - Go-live of the blockchain system planned for 2021 / 2022

- **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

- **Lygon.io**
  - Joint initiative by Australian Banks
  - Platform for blockchain-based bank guarantees for commercial property leases
    - “Before Lygon, issuing a paper bank guarantee took up to a month. Today, Lygon achieves same-day issuance.”
  - Digital bearer instrument
Programmable Money [16]: core idea

- Conditional payments: the transfer money only when predefined rules hold.
  - Examples: welfare payments, employee expenses, insurance payouts, ...
- Traditionally: conditions are checked (manually) in reimbursement or pre-approval/audit processes
  - Violation of policies: no reimbursement (or similar)

- Programmable money: next-generation conditional payments, on decentralized ledger / blockchain.
- In our programmable money project: programmed policies are not attached to accounts, but instead to money itself!
  - Policies here specify under which conditions money may be spent
  - When you try to spend money, the **money itself checks** automatically if a payment adheres to the policies
  → no uncertainty whether you will get reimbursed (and other benefits)
Programmable Money (”making money smart”)

Use case: National Disability Insurance Scheme (NDIS)

**Tokens** represent value of AUD for NDIS purchases.

**Pouches** represent different quantities of tokens.

**Policy contracts** stipulate rules and enforcements (e.g. ownership, eligible services, nominations, etc).

**Smart tokens** are formed when policy contracts are attached to pouches. Contracts can be destroyed when no longer required (e.g. after payment).

**Provider Registry Contract**

Providers are registered on a large policy contract.

**Participant plans**

Participant plans have pouches of smart tokens for different budgets, which can be spent on services from providers.

**Service Agreement Contracts**

Service agreements provisionally attach tokens to providers and enable payments as services are delivered.
Programmable Money: our NDIS proof of concept

- **NDIA**: Blockchain tokens reflect plan budgets
- **Participant**: Policy contracts reflect budget rules
- **Conditions checked**: Participant books eligible services using the app
- **Service provider**: Service providers receive smart tokens for eligible services
- **Policy contracts can blend plan management approaches**: Carers / Guardians, Plan Manager, Agency Manager

**New Payments Platform**
- Service provider redeems smart tokens for payment
- NDIA facilitates data-rich payments in near real-time
Programmable Money: further notes

• Many more details contained in the keynote paper

• Including lessons learnt and some open questions, for programmable money and development of blockchain apps in general. Examples:
  • How to present the policies in a way that the users can understand them?
  • How to horizontally scale components that create and submit transactions on behalf of a single party?
Selected blockchain adoption examples

https://www.unicef.org/innovation/blockchain
Summary

• Blockchain basics and terminology
• Designing and developing blockchain applications
  • Architecture & design
  • Model-driven engineering
• Blockchain and Services:
  • Integrating Blockchain-based Applications with Services
  • Blockchain-as-a-Service
  • Service-orientation vs. Smart Contracts
• Programmable money
• Blockchain adoption
Blockchain Application Design and Development, and the Case of Programmable Money

CLOSER’21 Keynote

Prof. Dr. Ingo Weber | April 2021

ingo.weber@tu-berlin.de | linkedin.com/in/ingomweber/ | Twitter: @ingomweber
References (1)


