Softwareentwicklung für Blockchains

Entwicklungsprozess einer DAPP in Theorie und Praxis

Arne Köller

Vortrag im Fach Softwaretechnik und Systemsoftware

Hochschule Bochum

Wintersemester 2017-2018

Vorabbemerkung

Bitcoin ≠ Blockchain

Ziele dieses Vortrages

1. Generelles Verständnis vom Konzept, Aufbau und Funktionsweise einer Blockchain, inbesondere von Ethereum.

2. Transfer der “klassischen” Software-Engineering-Fragestellungen auf Blockchain-Orientiertes Software-Engineering.

3. Einblick in die Praxis der Entwicklung von Smart Contracts und DAPPs.

Gliederung

Einführung in Blockchains

1. Relevanz des Themas
2. Historie der Blockchain
3. Konzept einer Blockchain
4. Einführung in Ethereum mit Demo

Software Engineering für Blockchains

1. Teilgebiete des Software Engineerings
2. Software-Engineering: Planung
3. Software-Engineering: Entwurf
4. Software-Engineering: Implementation
5. Software-Engineering: Test
6. Software-Engineering: Release und Wartung
7. Software-Engineering: Projektmanagement & QM
8. Development-Pipleline mit *truffle*
9. Praktische Umsetzung mit *truffle*

Abschluss

1. Ausblick
2. Zusammenfassung
3. Quellen
4. Diskussion und Fragen

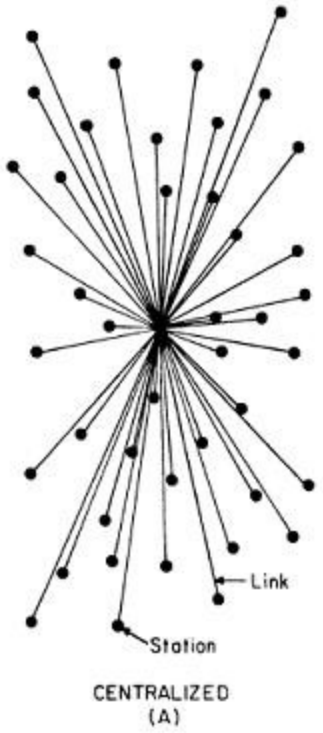
Einführung in Blockchains

1. Relevanz des Themas
2. Historie der Blockchain
3. Konzept einer Blockchain
4. Einführung in Ethereum mit Demo

Einführung in Blockchains

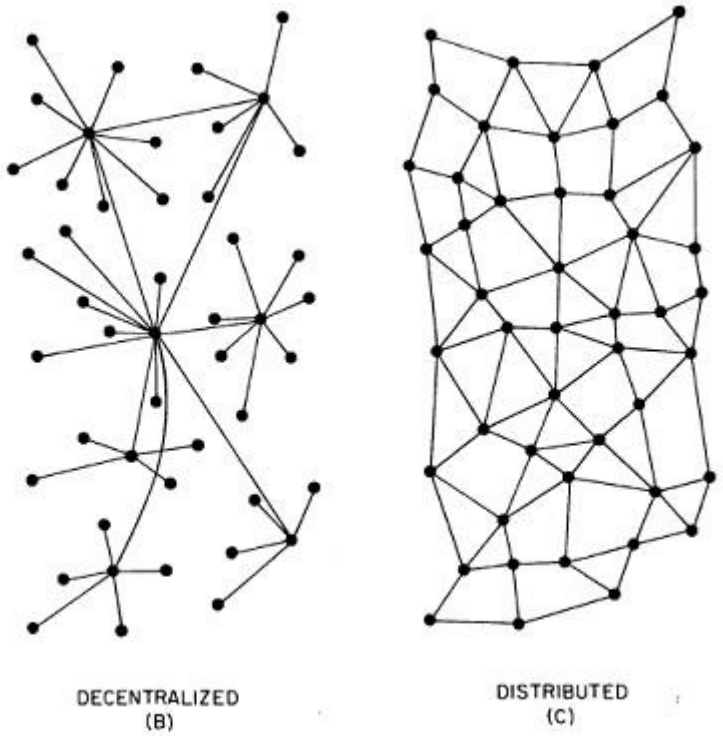
Relevanz des Themas Blockchain

Web 2.0



Google, Facebook, Amazon, Paypal

Web 3.0



Bitcoin, Ethereum, IOTA

Quellen: Siraj Raval
URL: https://github.com/llSourcell/Your_First_Decentralized_Application/blob/master/
Stand: 09. Januar 2018.

Relevanz des Themas Blockchain

Jetzt betrachten wir die genannten Netztopologien hinsichtlich der folgenden Aspekte:

- Integrität
- Verfügbarkeit
  - Ausfallsicherheit
  - Zensur
  - Zugangsbedingungen
- Vertraulichkeit

} Die drei klassischen Ziele der Informationssicherheit

Relevanz des Themas Blockchain

Ziel	Zentralisiert	Dezentralisiert/Verteilt
Integrität der Daten	Anhängig vom Pflegeaufwand des Betreibers	Durch komplexen Mining-Algorithmus sind Daten unveränderbar.
Ausfallsicherheit	Server des Betreibers als Single-Point-Of-Failure.	Jeder Teilnehmer (Node) müsste ausfallen.
Zensur	Betreiber kann gezwungen werden Daten zu zensieren.	Jeder Teilnehmer hat alle Daten.
Zugangsbedingungen **(!)**	If you are not paying for the product, you are the product.	Trustless-System ohne Gebühren und Datenkraken.
Vertraulichkeit der Daten	Abhängig vom Sicherheitsaufwand des Betreibers.	Daten durch Public-Key anonymisiert, jedoch sollten **NIE** stark personenbezogene Daten gespeichert werden.

Relevanz des Themas Blockchain

(!)

Dezentrale/verteilte Architektur dann nutzen, wenn Daten offen und ohne Mittelsmann übertragen werden soll.

- Eine Blockchain implementiert eine solche dezentrale/verteilte Architektur!

Beispiele:

- Bitcoin soll Banken überflüssig machen.
- Smart Contracts (Ethereum) sollen Treuhandgeschäfte wie airbnb überflüssig machen.

Historie der Blockchains

- Der Japaner Satoshi Nakamoto began 2007 an der Formulierung und Implementierung der ersten Blockchain mit dem Ziel eine dezentraliserte Digitalwährung zu schaffen.
- Am 31. Oktober 2008 veröffentlichte Nakamoto sein White-Paper zu Bitcoin unter dem Titel “Bitcoin: A Peer-to-Peer Electronic Cash System”.
- Am 3. Januar 2009 wurde der Ur-Bitcoin-Block erstellt.
- Am 9. Januar 2009 ging Bitcoin v0.1 live. [1]
- Vitalik Buterin formulierte Ende 2013 erstmals die Idee von Ethereum. [2]

Historie der Blockchains

- Im May 2015 wurde die erste Ethereum-Version “Olympic” veröffentlicht. [2]
- Sønstebø, Ivancheglo, Schiener und Popov gründen 2015 die Next-Generation-Blockchain “IOTA”. [3]
- Heute (09.01.2018 16:30 MEZ) ist ein Bitcoin umgerechnet 12.085,95 Euro wert. [4]

Konzept einer Blockchain

- Aufgabe Nakamotos: Lösung des Double-Spending-Problems in einem dezentralisierten System.

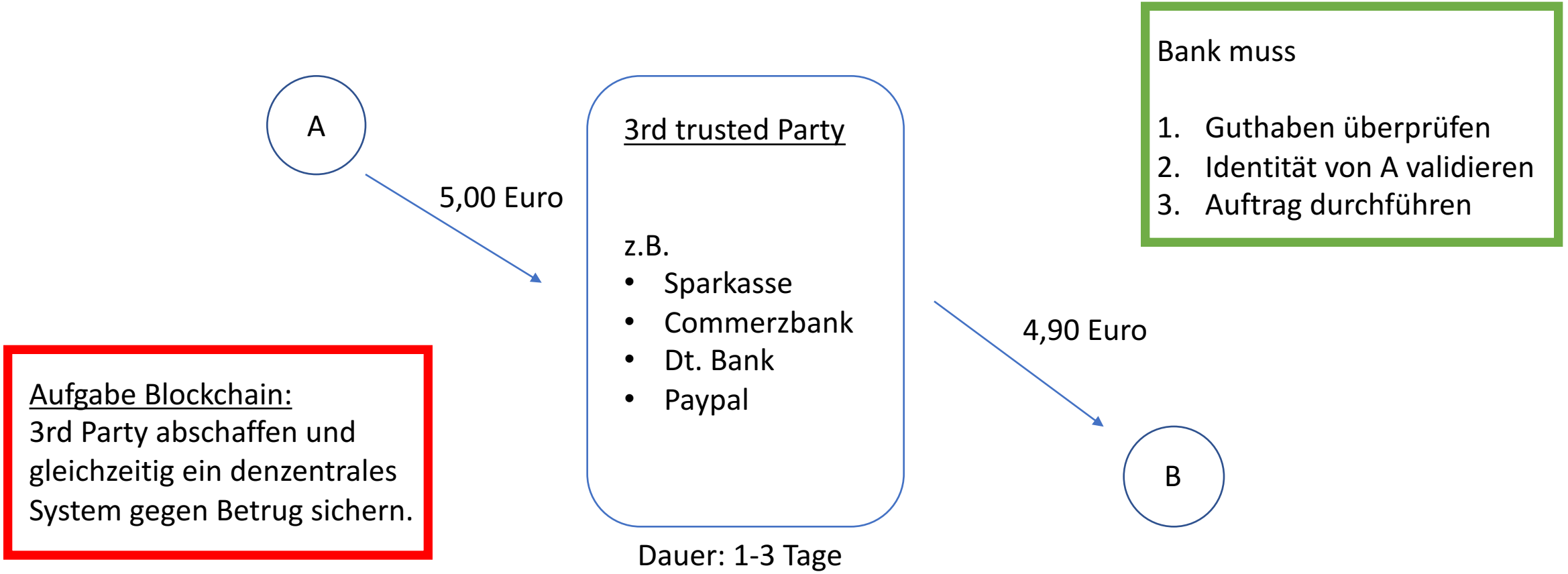
- Blockchains wurde von Nakamoto erdacht, um einen mittelsmannlosen, digitalen Geldtransfer zu ermöglichen.

- Intuitiv sind Blockchains am besten an Geldtransfers zu erklären.

- Eine solche Blockchain-Plattform besteht aus zwei Entitäten:
  1. Den Accounts bzw. Konten.
  2. Einer Blockchain, die die Transaktionen zwischen Konten aufzeichnet.

Konzept einer Blockchain

Zentralisierter Ansatz am Beispiel Geldtransfer



Konzept einer Blockchain

Das Blockchain-Rezept:

Open Ledger (Offenes Buch)

- Jeder kann alle Transaktionen einsehen.

\+ Distributed Ledger (Verteiltes Buch)

- Jeder hat eine Kopie aller Transaktionen lokal vorliegen.

\+ Mining/Consensus-Algorithms (Synchronisation des Buches)

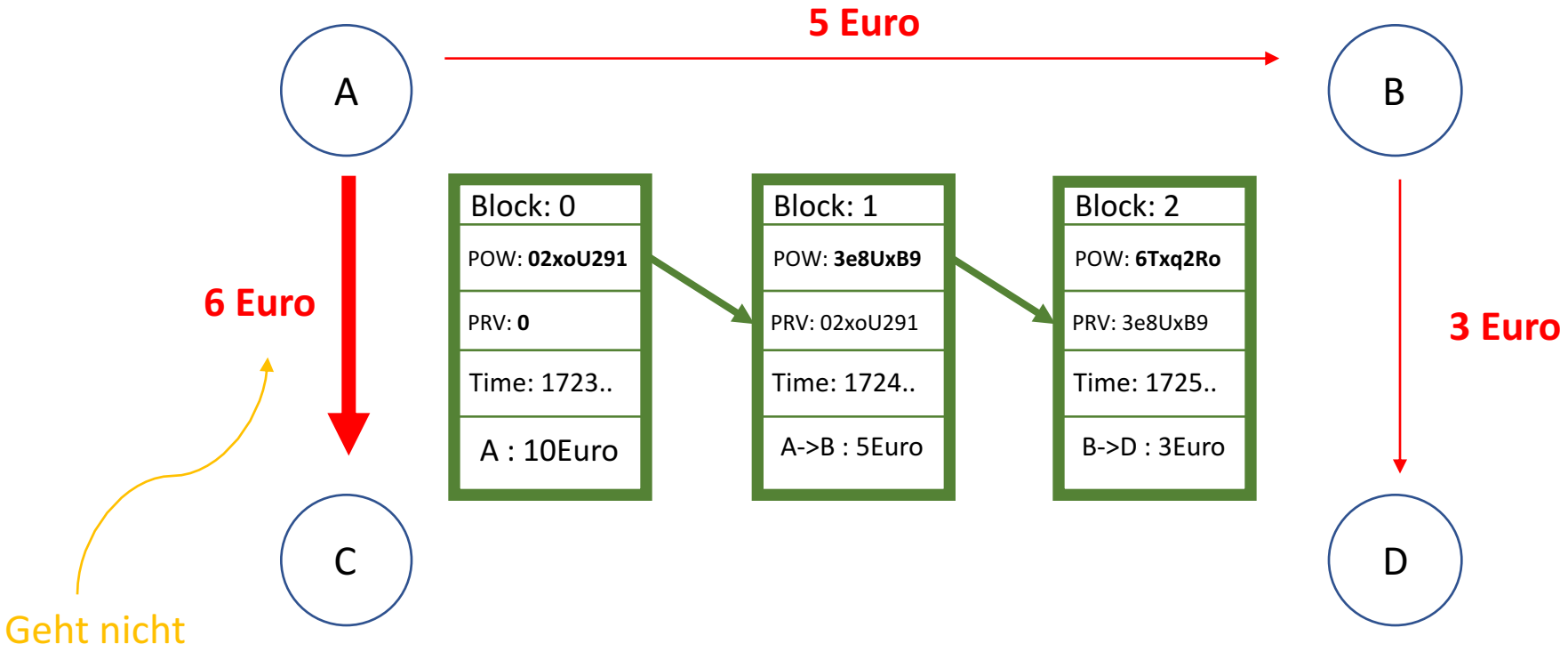
- Miner validieren und synchronisieren neue Transaktionen.

= Blockchain

Konzept einer Blockchain

Dezentralisierter Ansatz am Beispiel Geldtransfer

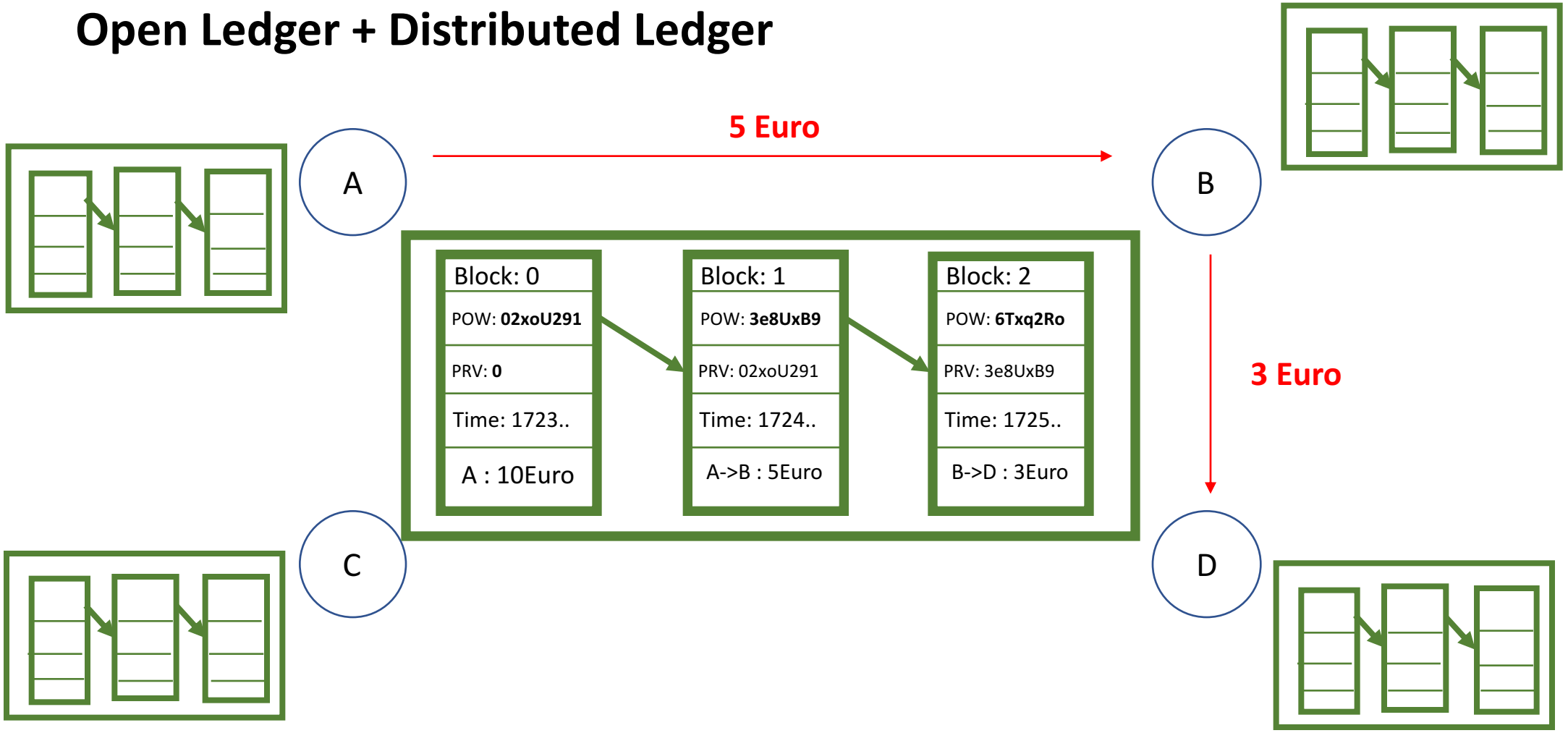
Open Ledger



Konzept einer Blockchain

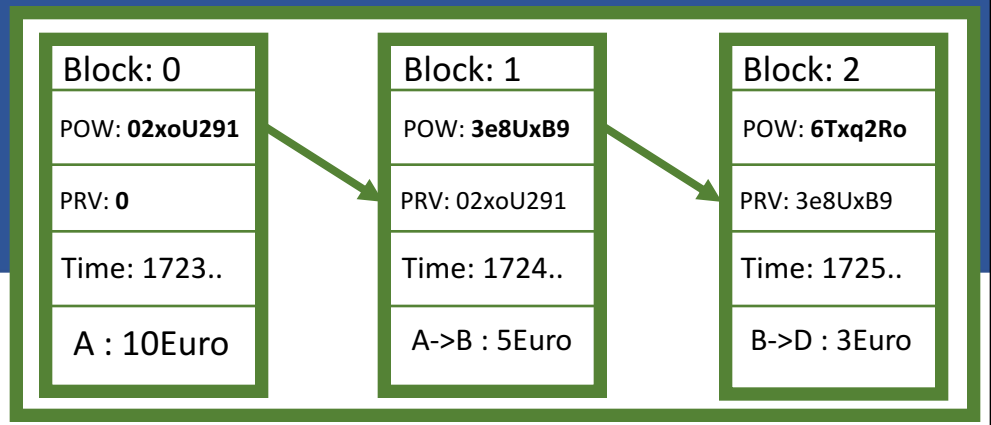
Dezentralisierter Ansatz am Beispiel Geldtransfer

Open Ledger + Distributed Ledger



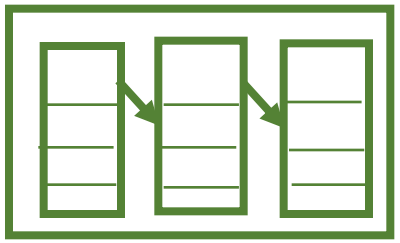
Einführung in Blockchains

Konzept einer Blockchain



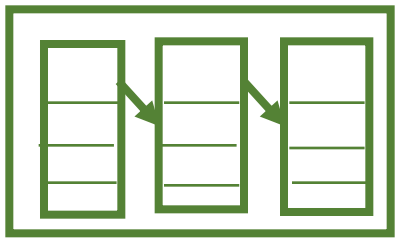
Dezentralisierter Ansatz am Beispiel Geldtransfer

Open Ledger + Distributed Ledger + Mining



A

5 Euro

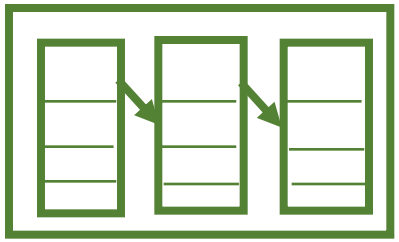


C

Miner-Algorithmus

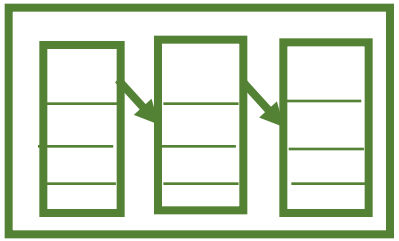
1. Publish transaction on network to miners
2. Miners validate invalidated transaction
   - 2.1 Check if balances okay
   - 2.2 Check if sender authenticated
   - 2.3 Find Proof-Of-Work (POW)
   - 2.4 Collect reward
   - 2.5 Broadcast validated transaction
3. Every other node checks POW
4. If >50% of nodes agree, they insert a new transaction-block

B



Block: 3
POW: **rM7bxa9**
PRV: 6Txq2Ro
Time: 1726..
A->C : 5Euro

D



Betrug benötigt >50% Mineranteil + alle POWs neu berechnen! Dies löst das Double-Spending-Problem.

Konzept einer Blockchain

Eine Blockchain hat somit die folgenden Eigenschaften:

- Offen/Transparent (durch Open Ledger)
  - Jeder Teilnehmer kann alle Transaktionen einsehen.
- Dezentralisiert (durch Distributed Ledger)
  - Jeder Teilnehmer hat eine Kopie aller Transaktionen.
- Immutable and Irreversible (durch Mining)
  - Unveränderbare Blöcke und Block-Reihenfolge.

Blockchains werden immer dann nützlich, wenn Daten offen, dezentral und unveränderbar gespeichert werden sollen.

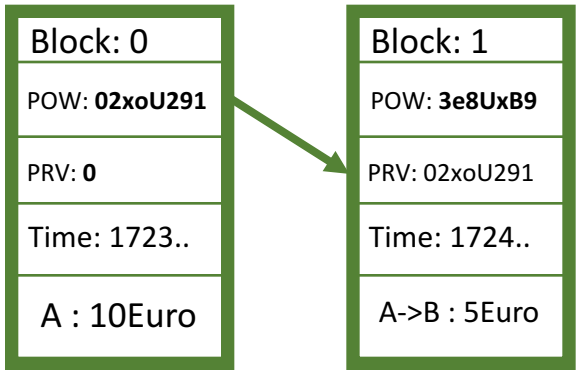
- Z.B. bei Prüfungsnoten, Gesundheitsdaten, Wahlstimmen, Supply-Chain, Product-Chain, Geldtransfers, Besitzurkunden, Aktienanlagen, Verwaltungsvorgängen (Governance), Paket- und Brieflogistik, Maschinendaten, etc.

Einführung in Ethereum

- Ethereum ist eine offene, dezentrale Blockchain-Plattform mit der Smart Contracts programmiert werden können. [5]
- Ethereum kennt zwei Account-Typen: User-Accounts (EOA) und Contract-Accounts. [5]
- Ein Smart Contract “lebt” nur in einem Contract-Account und ist in der Ethereum-Programmiersprache Solidity geschrieben.
- Ein Smart Contract ist wie eine Java/C++-Klasse: Es gibt Klassenvariablen (Status) und Methoden (calls, transaktionen).

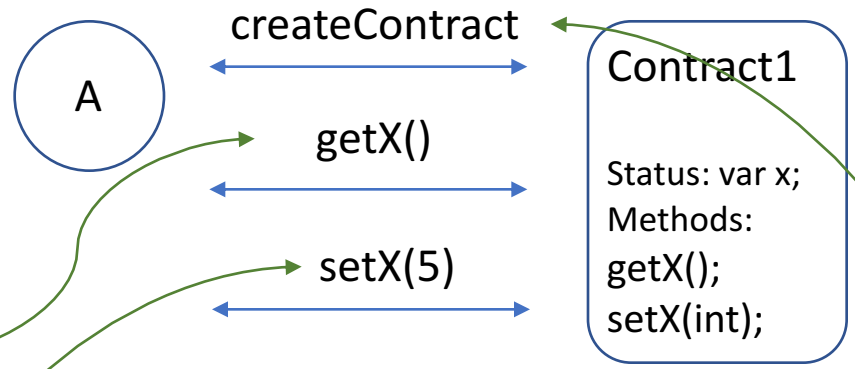
Einführung in Ethereum

Bitcoin-Blockchain

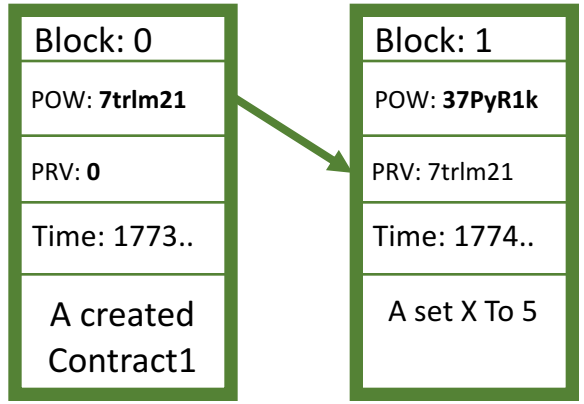


Transaktionen sind immer an Statusänderungen gekoppelt und werden in die Blockchain eingefügt.

Ethereum-Blockchain



calls werden nicht in die Blockchain eingefügt.



Wird in Blockchain eingefügt (Vertrag nur gültig, wenn er der Blockchain bekannt ist).

Einführung in Ethereum

Ethereum setzt erst einmal die gleichen Grundkonzepte einer Blockchain um, wie eine Geldtransfer-Blockchain.

Open Ledger (Offenes Buch)

- Jeder kann alle Transaktionen zwischen EOA und Contracts einsehen.

\+ Distributed Ledger (Verteiltes Buch)

- Jeder hat eine Kopie aller Transaktionen zw. EOA und Contracts lokal vorliegen.

\+ Mining/Consensus-Algorithms (Synchronisation des Buches)

- Miner validieren und synchronisieren neue Transaktionen.

= Blockchain

Mit einem Zusatz:

Einführung in Ethereum

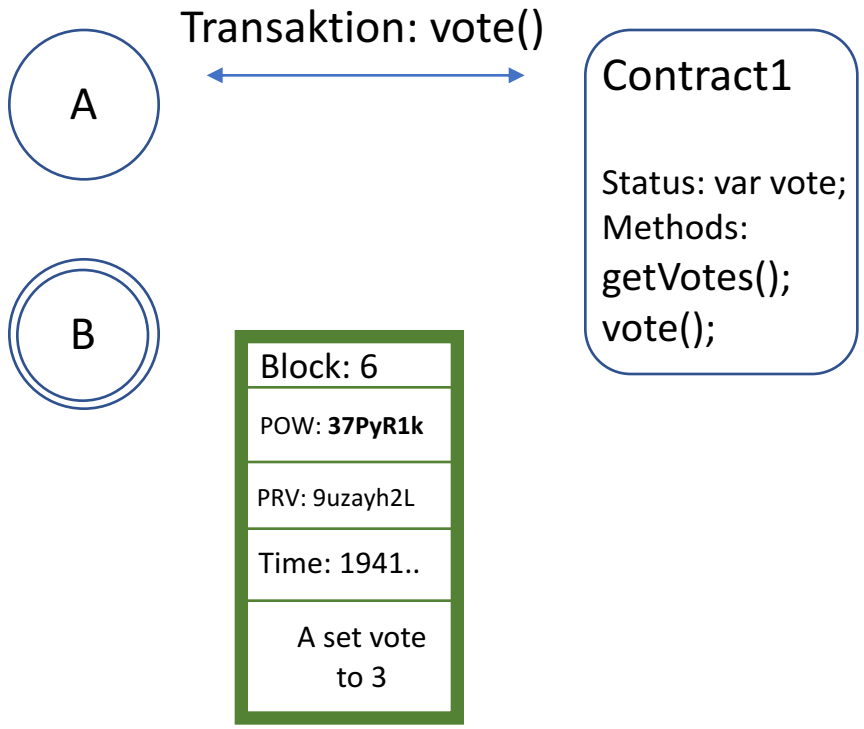
Bitcoin-Blockchain vs. Ethereum-Blockchain

Bei einer Geldtransfer-Blockchain gibt es nur einen einheitlichen Transaktionstypen, bei einer programmierbaren Blockchain können eigene Transaktionstypen bestimmt werden.

Einführung in Ethereum

Frage: Was haben Smart Contracts für einen Vorteil gegenüber herkömmlichen Verträgen?

Szenario:



Mining-Algorithmus Ethereum

1. Publish transaction intend to all miners
2 . Miners validate invalidated transaction
    - 2.1 Check if sender authenticated
    - 2.2 Get Smart-Contract Code from Contract
    - 2.3 Execute code on **Ethereum Virtual Machine**
    - 2.4 Write Result in Data-Field of new Block
    - 2.5 Find Proof-Of-Work (POW)
    - 2.6 Collect reward for code execution
    - 2.7 Broadcast validated transaction
3. Every other node checks POW
4. If >50% of nodes agree, they insert a new transaction-block

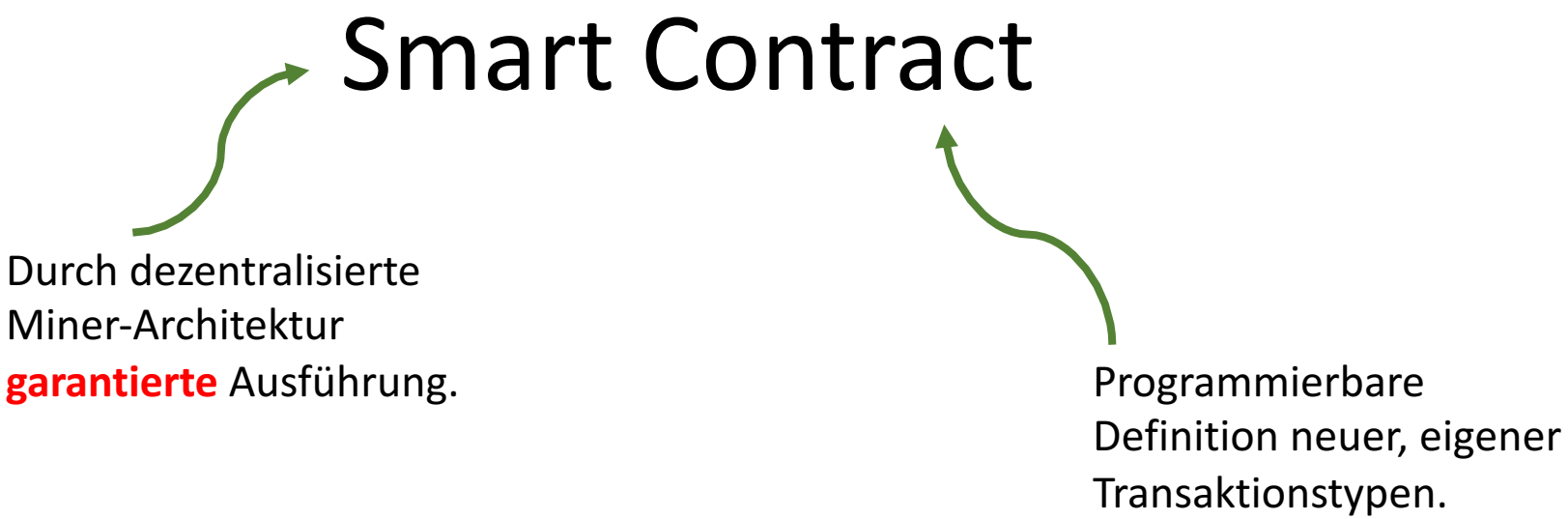
Einführung in Ethereum

Frage: Was haben Smart Contracts für einen Vorteil gegenüber herkömmlichen Verträgen?

Antwort:

- Durch Kopplung des Minings an die Codeausführung eines Smart Contracts ist sichergestellt, dass der Vertrag erfüllt wird.
- Smart Contracts und Mining ersetzen somit Mittelsmänner wie z.B. Treuhänder, Gerichte, Verwaltungen, Banken, etc.

Einführung in Ethereum



Einführung in Ethereum

Beispiel für einen sinnvollen Smart Contract

1. Alice agrees to store her payment for the patio within the escrow contract, and Bob agrees to deposit an equal amount

2. Bob completes the patio project and Alice gives the smart contract permission to release the funds

3. Bob receives Alice's payment along with his collateral

Quelle: http://truffleframework.com/tutorials/ethereum-overview
Stand: 09. Januar 2018.

Einführung in Ethereum

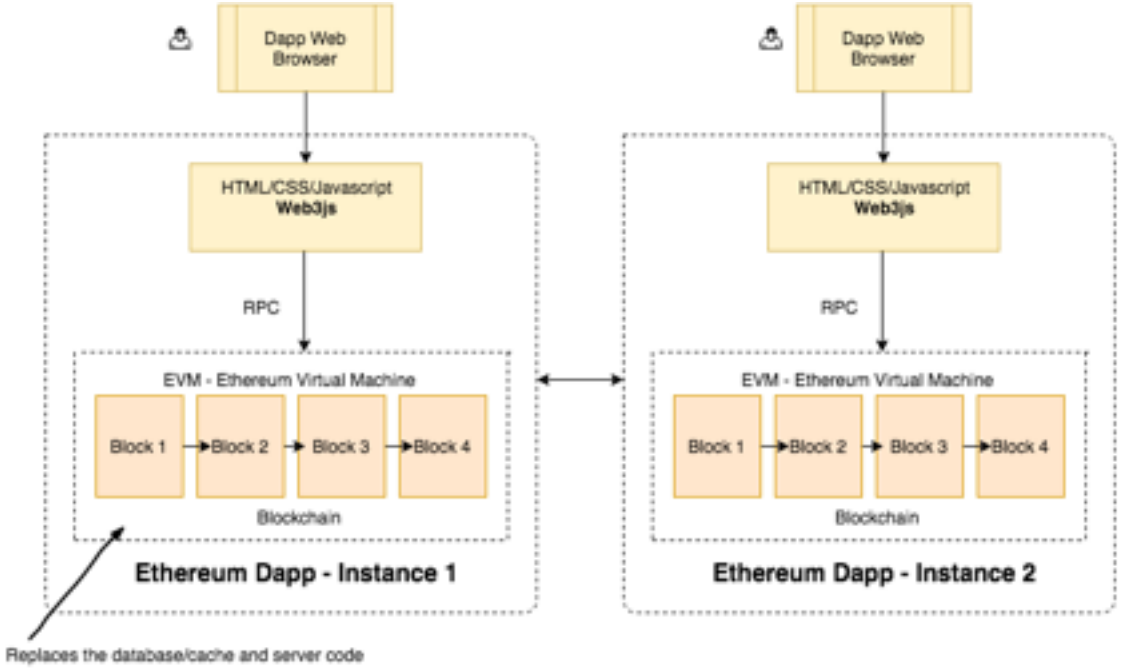
Ethereum Demo mit der Remix-IDE und Ganache

Software Engineering für Blockchains

1. Teilgebiete des Software Engineerings
2. Software-Engineering: Planung
3. Software-Engineering: Entwurf
4. Software-Engineering: Implementation
5. Software-Engineering: Test
6. Software-Engineering: Release und Wartung
7. Software-Engineering: Projektmanagement & QM
8. Development-Pipleline mit *truffle*
9. Praktische Umsetzung mit *truffle*

Teilgebiete des Software Engineerings

Frage zunächst: Was ist eine dApp?

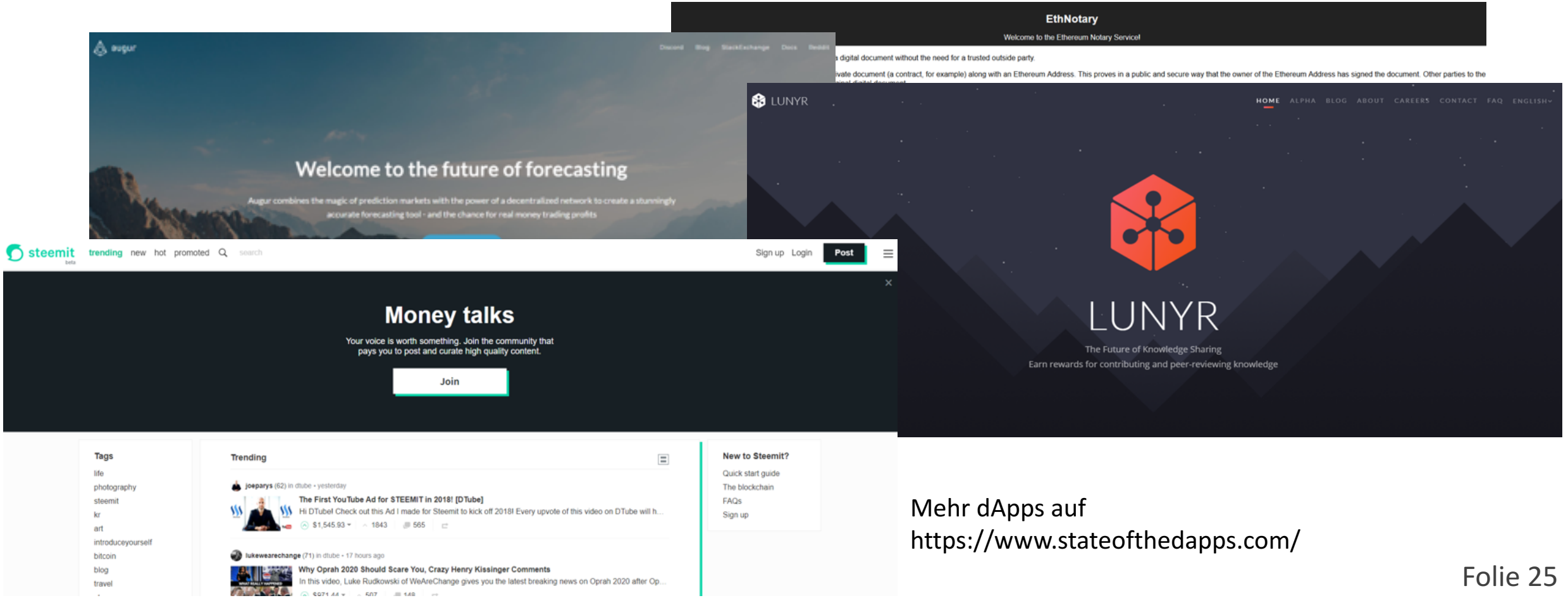


Eine dApp ist eine „Decentralized Application“, also eine App, deren Back-End auf einer programmierbaren Blockchain basiert und zusätzlich ein Front-End besitzt.

Quelle: Siraj Raval.
URL: https://github.com/llSourcell/Your_First_Decentralized_Application/blob/master/A%20Guide%20to%20Building%20Your%20First%20Decentralized%20Application.ipynb
Stand: 09. Januar 2018.

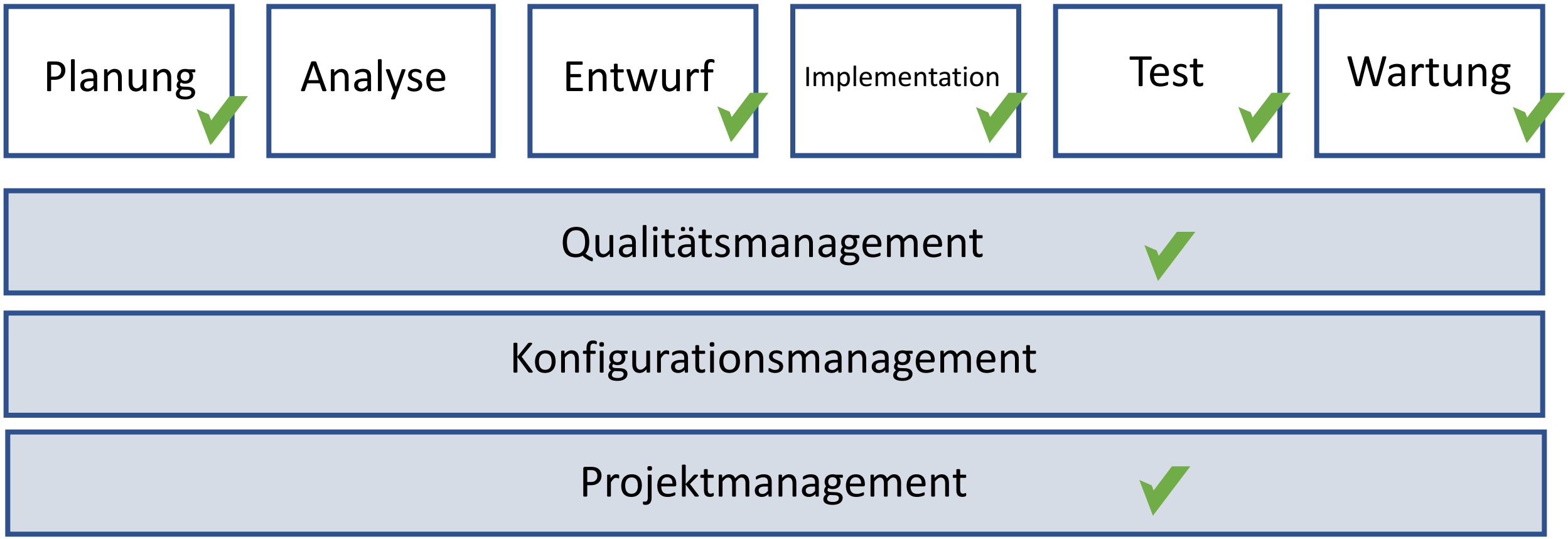
Teilgebiete des Software Engineerings

Es gibt bereits eine Vielzahl von Ethereum-dApps!



Mehr dApps auf
https://www.stateofthedapps.com/

Teilgebiete des Software Engineerings



Software-Engineering: Planung

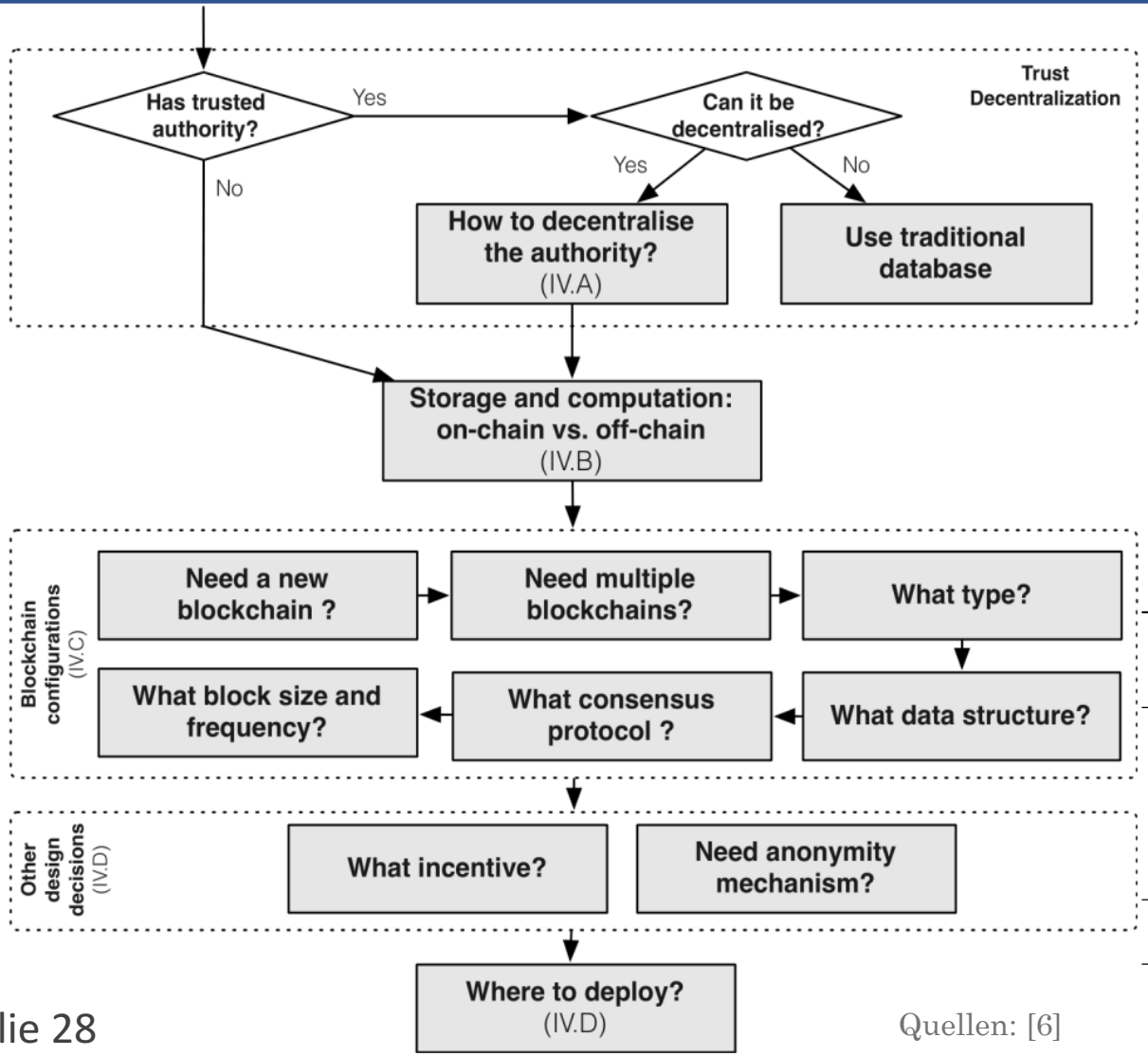
- Frage: Gibt es einen “Roten Faden” für die Planung von Blockchain-Applikationen/dApps?

- Viele Entwickler sind noch unerfahren auf diesem Gebiet, müssen aber z.B. schon ein Lastenheft erstellen.

- Hier kann man auf Forschungsergebnisse zurückgreifen und einen empirischen Entscheidungsbaum nutzen. [6]

Software Engineering für Blockchains

Software-Engineering: Planung



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

Design Decision	Option	Impact: Fundamental properties	Cost efficiency	Performance	#Failure points
Fully Centralised	Services with a single provider (*e.g.*, governments, courts)	⊕	⊕⊕⊕	⊕⊕⊕	1
	Services with alternative providers (*e.g.*, banking, online payments, cloud services)				
Partially Centralised & Partially Decentralised	Permissioned blockchain with permissions for fine-grained operations on the transaction level (*e.g.*, permission to create assets)	⊕⊕	⊕⊕	⊕⊕	*
	Permissioned blockchain with permissioned miners (write), but permission-less normal nodes (read)				
Fully Decentralised	Permission-less blockchain	⊕⊕⊕	⊕	⊕	Majority (nodes, power, stake)

		Fundamental properties	Cost efficiency	Performance	#Failure points
Verifier	Single verifier trusted by the network (external verifier signs valid transactions; internal verifier uses previously-injected external state)	⊕⊕	⊕⊕	⊕⊕	1
	M-of-N verifier trusted by the network	⊕⊕⊕	⊕	⊕	M
	Ad hoc verifier trusted by the participants involved	⊕	⊕⊕⊕	⊕⊕	1 (per ad hoc choice)

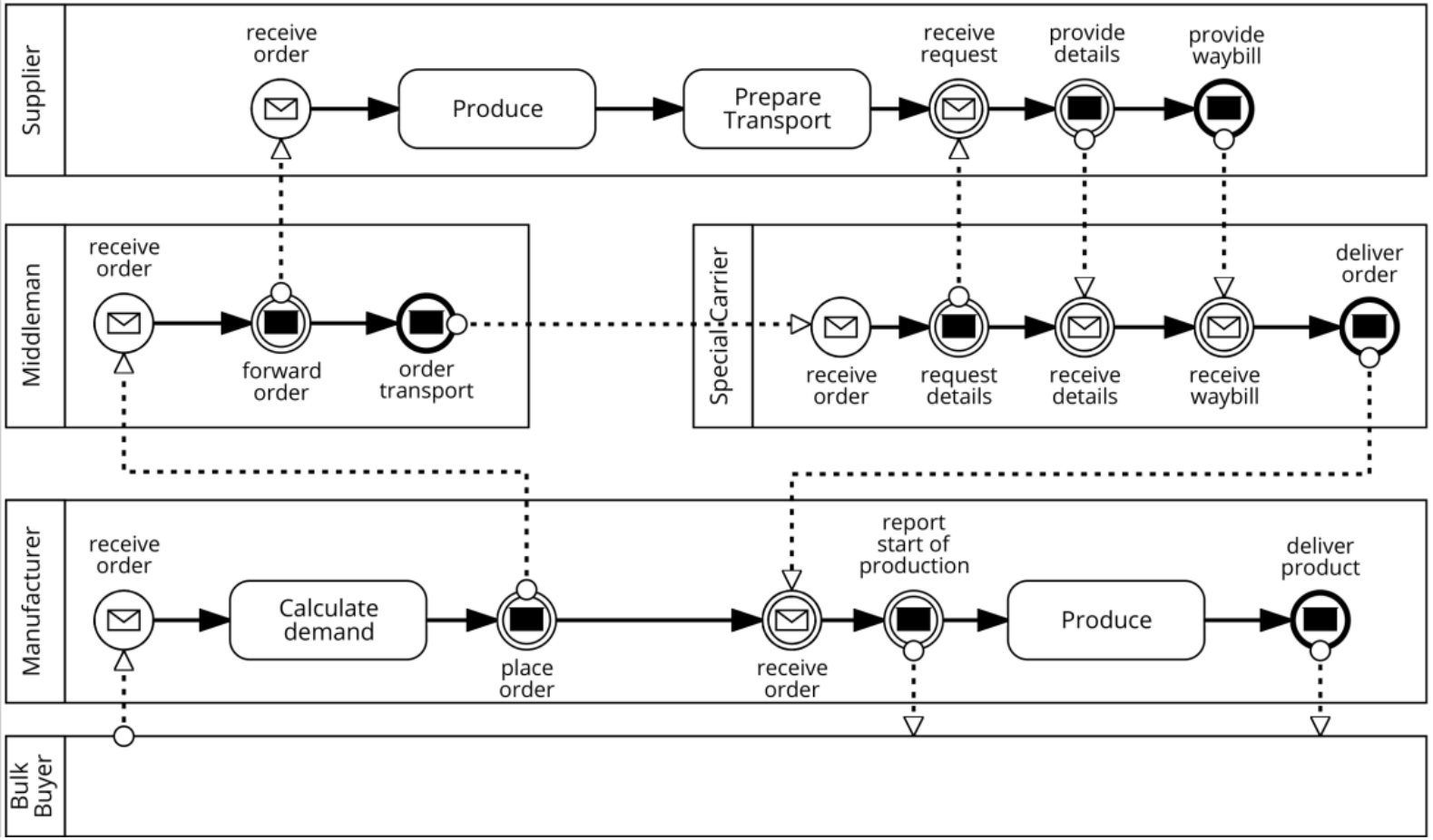
Design Decision		Option	Impact: Fundamental properties	Cost efficiency	Performance	Flexibility
Item data	On-chain	Embedded in transaction (Bitcoin)	⊕⊕⊕⊕	⊕	⊕	⊕⊕
		Embedded in transaction (Public Ethereum)		⊕⊕⊕⊕	⊕	⊕⊕⊕
		Smart contract variable (Public Ethereum)		⊕⊕	⊕⊕⊕	⊕
		Smart contract log event (Public Ethereum)		⊕⊕⊕	⊕⊕	⊕⊕
	Off-chain	Private / Third party cloud	⊕	~KB Negligible	⊕⊕⊕⊕	⊕⊕⊕⊕
		Peer-to-Peer system		⊕⊕⊕⊕	⊕⊕⊕	⊕⊕⊕
Item collection	On-chain	Smart contract	⊕⊕⊕⊕	⊕⊕⊕⊕ (public)	⊕⊕⊕⊕	⊕
		Separate chain		⊕ (public)	⊕	⊕⊕⊕⊕
Computation	On-chain	Transaction constraints	⊕⊕⊕⊕	⊕	⊕	⊕
		Smart contract				
	Off-chain	Private / Third party cloud	⊕	⊕⊕⊕⊕	⊕⊕⊕⊕	⊕⊕⊕⊕

Quellen: [6]

Software-Engineering: Entwurf

- Frage: Wie kann ich eine Blockchain-Applikation/dApp modellieren?
- Klassisch: UML-Klassendiagramm, UML-Aktivitätsdiagramm, UML-Interaktionsdiagramm, etc. Für dApp nicht möglich! [7]
- Wieder kann ein Entwickler sich an angewandter Forschung orientieren! [6]

Software-Engineering: Entwurf



Quelle: [6]

<u>Ziel:</u>
Business-Process mittels Smart Contracts umsetzen.

Smart Contracts dienen hier zu:

1. Tracker und Monitor
2. Treuhänder und Bezahlung
3. Business Integration Protokoll
4. Verschlüsselung

Es wird aber auch klar:
Physische Wertschöpfung kann eine Ethereum-Blockchain nicht ersetzen. *

* Die IOTA-Blockchain kann diese aber abstrahieren!

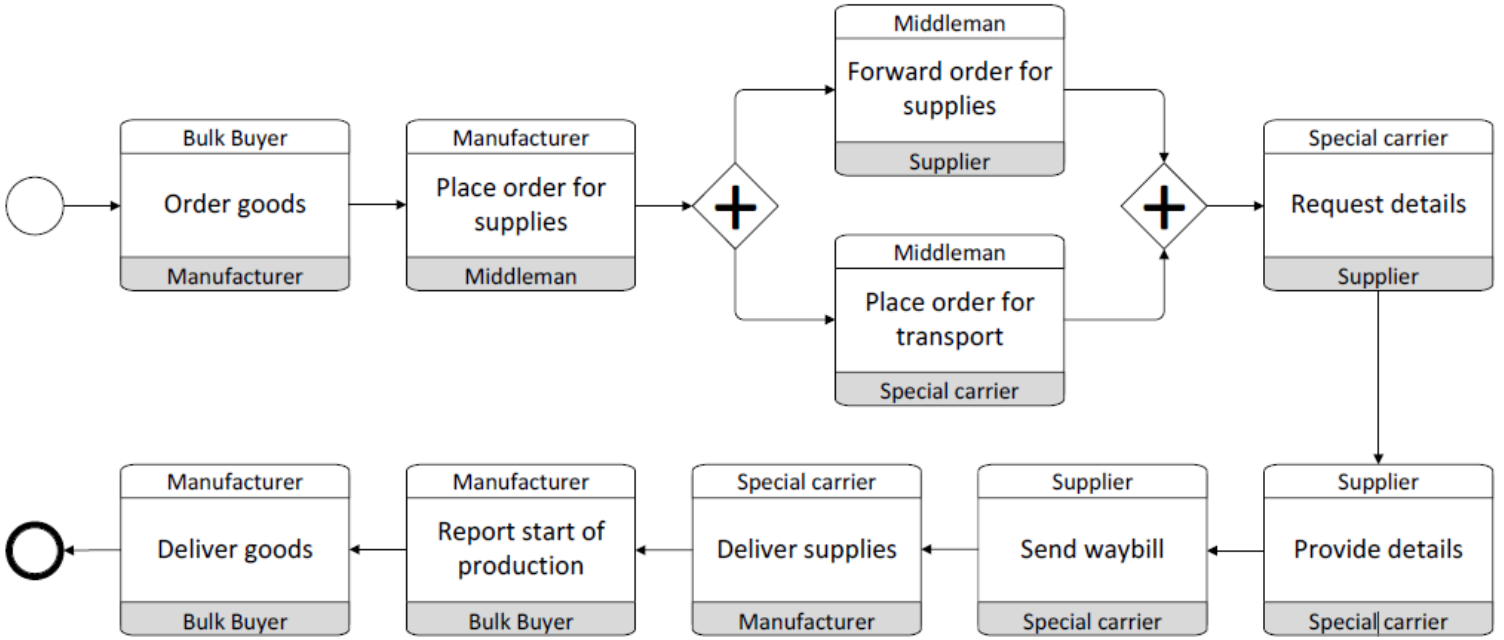
Software-Engineering: Entwurf

Vorgeschlagene Schritte zur Umsetzung in einen Smart Contract: [6]

1. BPMN aufstellen.
2. BPMN-to-Smart-Contract-Translator nutzen.
3. Smart Contract mit Außenwelt und Front-End-GUI verbinden.
4. Interaktionsdiagramm zw. Smart Contract und Außenwelt erstellen.

Software-Engineering: Entwurf

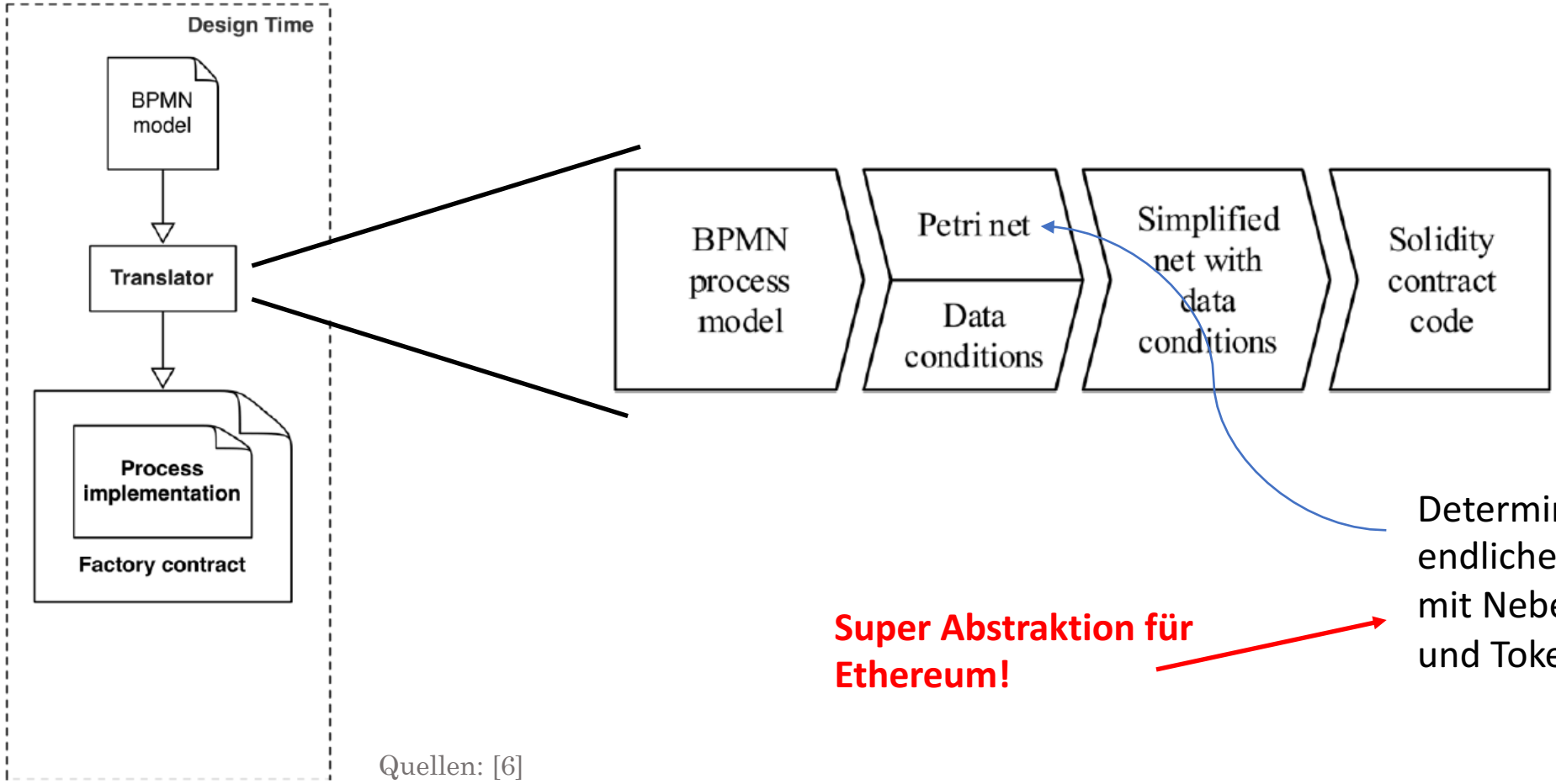
1. BPMN aufstellen.



Quelle: [6]

Software-Engineering: Entwurf

2. BPMN-to-Smart-Contract-Translator nutzen.



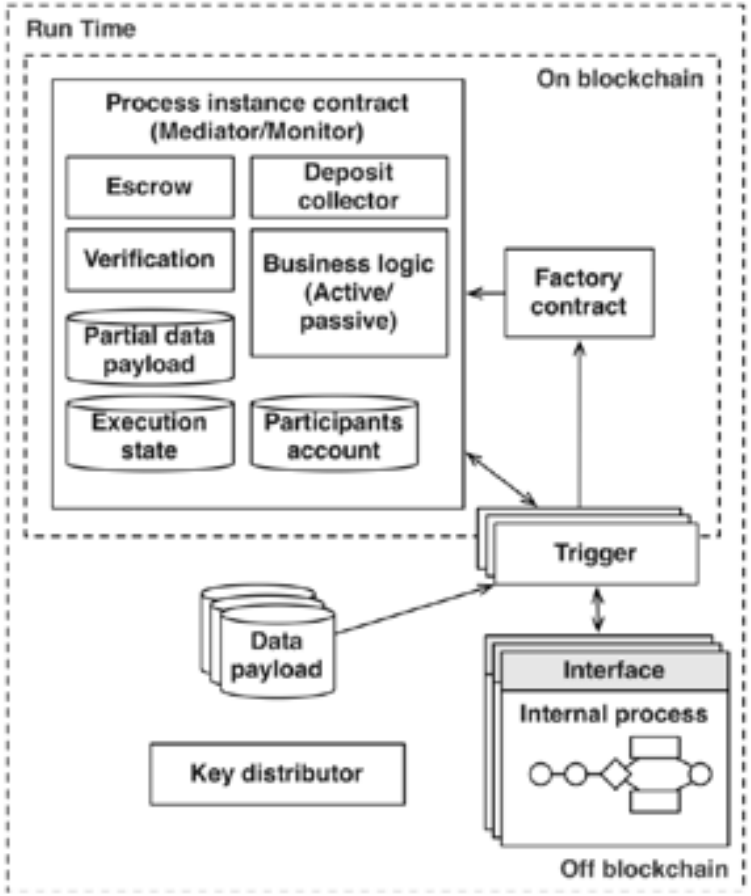
Deterministischer, endlicher Automat mit Nebenläufigkeit und Tokenkonzept

Super Abstraktion für Ethereum!

Quellen: [6]

Software-Engineering: Entwurf

3. Smart Contract mit Außenwelt und Front-End-GUI verbinden.

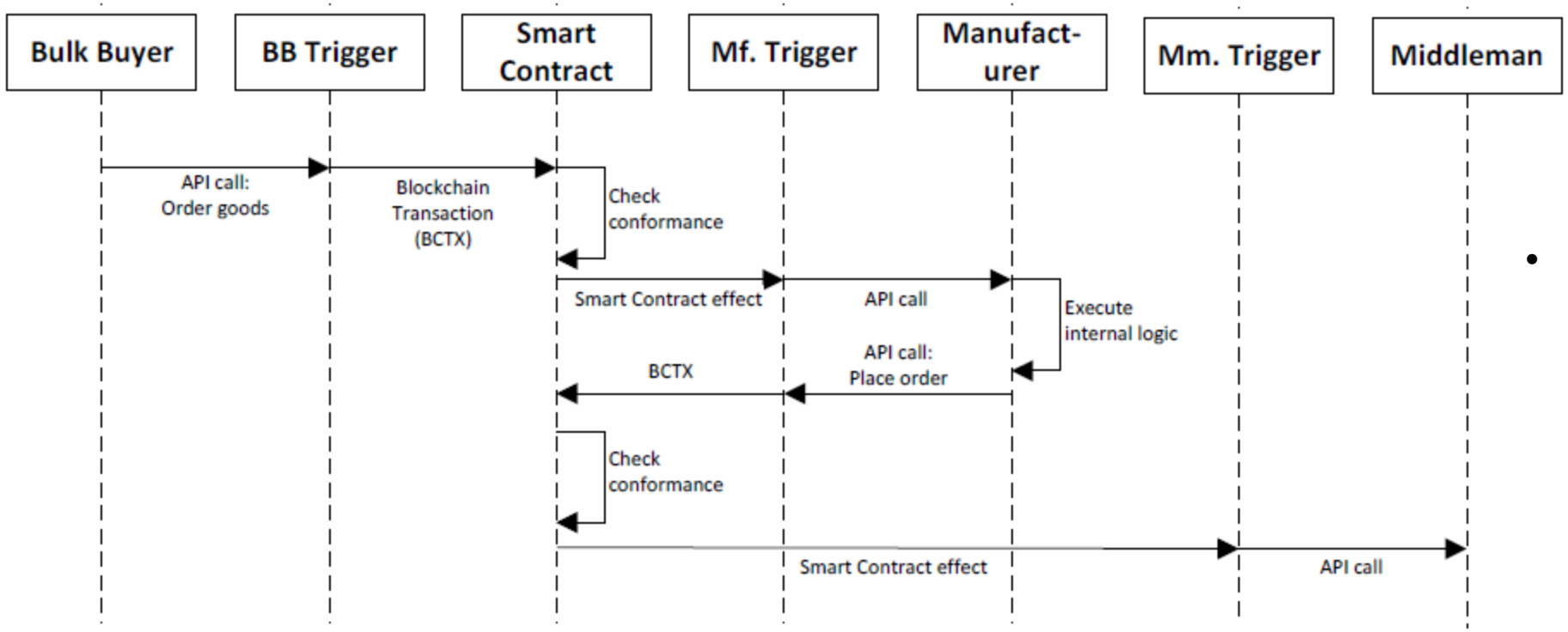


Quelle: [6]

- Aufteilung in On-Blockchain-und Off-Blockchain-Komponenten.
- Invokation über Trigger
- Instanziierung eines Smart Contracts pro Geschäftsprozess (also immer neuen Smart Contract in der Blockchain erstellen, wie in Remix)

Software-Engineering: Entwurf

4. Interaktionsdiagramm zw. Smart Contract und Außenwelt erstellen.



- Insb. für den Off-Blockchain-Applicationflow wichtig!

Quelle: [6]

Software-Engineering: Implementation

- Erstellung der Smart Contracts mit einem Translator (den gibt es übrigends nur in der Theorie bisher).
- Es sollte nie im Ethereum-Mainnet programmiert werden, sondern immer auf Test-Chains.
- Vorschläge für Design-Pattern in [9].
  - Es wird gezeigt, dass die Design-Pattern “Abstract Factory”, “Flyweight”, “Proxy” und “Publish-Subscribe” auch in einer Blockchain-Anwendung sinnvoll umgesetzt werden können.
- Gängiste Tools für die Entwicklung: Remix, VS Code Solidity-Plugin, geth (go-Ethereum-Client), node.js .
- Nutzung des dApp-Frameworks *truffle.*

Software-Engineering: Test

- Blockchain/EVM-Implementation-Tests: [8]
  - Cleanroom-Software-Engineering (Robustheit zertifizieren)
  - Continuous-Software-Reviews
- Smart-Contract-Tests: [8]
  - Entsprechen Business-Process-Model?
  - Entsprechen geltendem Recht? [9]
- Transaction-Tests: [8]
  - Latenztests [6]
  - Integritätstests

Quelle: Financial Tribune
URL: https://financialtribune.com/sites/default/files/field/image/12_Cleanroom.jpg
Stand: 10.01.2018.

Software-Engineering: Release und Wartung

- Nutzung eines Deployment Scripts von Vorteil.
- Dieses stellt z.B. sicher, dass unveränderte Smart Contracts nicht doppeltmigriert werden.
- Wenig Hardcode (z.B. Strings) in Transaktionscode schreiben. [9]

Software-Engineering: Projektmanagement & QM

- Interdisziplinäres Team gefordert! [7] [8]
  - Ein Smart Contract ist auch rechtlich ein Vertrag. [8]
  - Er darf geltendes Recht nicht verletzen.
  - Gewerbsmäßige Rechtsberatung ist in Deutschland nur Rechtsanwälten vorbehalten! [8]
  - Blockchain-Team der Zukunft: Informatiker + Anwalt.

- Vorschlag für Qualitätsmetrik: Goal-Question-Metric-Methode (GQM) [7]

Development-Pipeline mit *truffle*

Praktische Umsetzung mit *truffle*

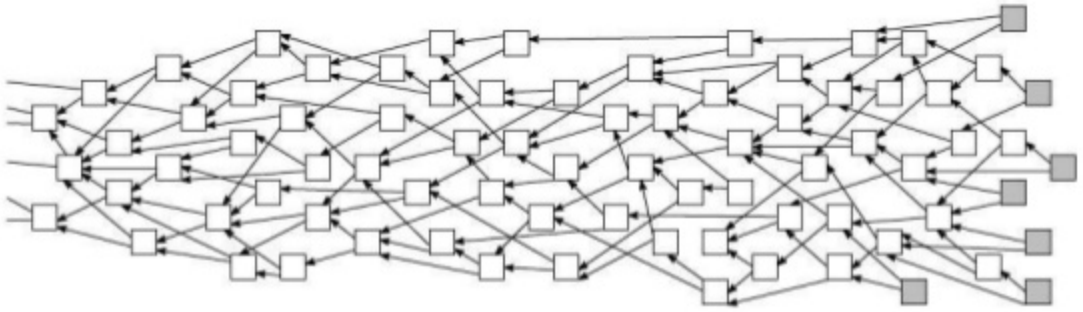
truffle Demo mit Ganache

Abschluss

1. Ausblick
2. Zusammenfassung
3. Quellen
4. Diskussion und Fragen

Abschluss
Ausblick

- Blockchains erst am Anfang ihrer Entwicklung.
- Ermöglicht nicht nur Geschäftsprozessoptimierung, sondern auch
  - effektive Korruptionsbekämpfung,
  - transparentere Demokratie,
  - Zurückgewinnung der Hoheit über eigene Daten,
  - gerechte Verteilung, Sharenomics und autonome Maschinenkommunikation.



IOTA Tangle
URL: https://siamblockchain.com/wp-content/uploads/2017/06/iota-ledger-of-things-18-638.jpg
Stand: 10.01.2018.

	Web 2.0	Web 3.0 (dApps)	Status
Scalable computation	Amazon EC2	Ethereum, Truebit	In progress
File storage	Amazon S3	IPFS/Filecoin, Storj	In progress
External data	3rd party APIs	Oracles (Augur)	In progress
Monetization	Ads, selling goods	Token model	Ready
Payments	Credit Cards, Paypal	Ethereum, Bitcoin, state channels, 0x	Ready

Das Web 3.0
Quelle: Siraj Raval
Stand: 10.01.2018.

Abschluss

Zusammenfassung

- Blockchains immer dann, wenn Bewegungsdaten offen, dezentral und unverändbar festgehalten werden sollen.
- Ethereum ist eine programmierbare Blockchain.
- Ethereum-Software-Engineering ist noch ein Forschungsfeld, aber es gibt erste Ansätze.

Quellen

[1] Quelle: Bitcoin Wiki.
URL: https://en.bitcoin.it/wiki/Category:History
Stand: 08. Januar 2018.

[2] Quelle: Ethereum Wiki.
URL: http://ethdocs.org/en/latest/introduction/history-of-ethereum.html
Stand: 08. Januar 2018.

[3] Quelle: Wikipedia Eintrag für “IOTA”.
URL: https://en.wikipedia.org/wiki/IOTA_(technology)
Stand: 08. Januar 2018.

[4] Quelle: Google Finance.
Stand: 09. Januar 2018.

Abschluss
Quellen

[5] Quelle: Ethereum Homestead Documentation.
URL: http://ethdocs.org/en/latest/introduction/what-is-ethereum.html
Stand: 09. Januar 2018.

[6] Author: Dr. Mark Staples.
Quelle: Vortrag “Software Engineering Research for Blockchain-Based Systems”.
Datum: 2017.

[7] Author: S. Porru, A. Pinna, M. Marchesi, R. Tonelli.
Quelle: “Blockchain-oriented Software Engineering: Challenges and New Directions”.
Datum: 2017.

[8] Author: Hajo Schulz, CT-Magazin für Computer Technik, Ausgabe 23/2017.
Quelle: Artikel “Vertrag denkt mit”, S. 108ff.
Datum: 28.10.2017.

Abschluss
Quellen

[9] Author: P. Zhang, J. White, D.C. Schmidt, G. Lenz.
Quelle: “Design of Blockchain-Based Apps Using Familiar Software Patterns to Address Interoperability Challenges in Healthcare”.
Datum: 2017.

Abschluss

Diskussion und Fragen

Z.B.:

- Wie funktionieren Mining-Algorithmen genau?
- Welche Blockchain-Technologie wird sich durchsetzen?
- Wo kann ich mehr über Blockchains lernen?
- Sollte ich jetzt ein Blockchain-Start-Up gründen?
- Welche Cryptowährung soll ich kaufen, um reich zu werden?

Vielen Dank.