Hochschule München

Digital Image Processing

Image Stitching and Point Cloud Generation from Smartphone Pictures

Verfasst von

Adam Mahmoud

Michael Wimmer

Philipp Kronser

Vorgelegt bei

Prof. Dr. Claudius Schnörr

22. Dezember 2018

Abkürzungsverzeichnis

BOM	Bill of Materials
BTC	Bitcoin
CAD	Computer-Aided Design
DApp	Dezentralisierte Apps
DRM	Design Research Methodology
ETH	Ether
ERP	Enterprise Ressource Planning
EVM	Ethereum Virtual Machine
PDM	Product Data Management
PLM	Product Lifecycle Management
PoS	Proof of Stake
PoW	Proof of Work
STEP	Standard for the exchange of product model data

Inhaltsverzeichnis

1. Einleitung

1.1. Motivation und Ziel der Arbeit

Lalala

1.2. Aufbau der Arbeit

Lalala

2. Deskriptive Phase I

2.1. Verwaltung von CAD-Daten heute

Am Computer konstruierte Bauteile werden mit sogenannten Computer-Aided Design (CAD) Programmen erzeugt. Dabei arbeiten viele Konstrukteure an einzelnen Bauteilen, welche nach Fertigstellung zu größeren Baugruppen zusammengefügt werden.

Zur Ablage und Verwaltung von CAD Daten kommen sogenannte Product Data Management (PDM)-Lösungen zum Einsatz. Grundsätzlich ist auch das lokale Abspeichern von CAD Daten möglich. Arbeiten jedoch mehrere Konstrukteure an einem Produkt, ist momentan eine zentrale Datenablage, auf die jeder zugreifen kann, unverzichtbar. Seit seiner Einführung in den späten 1980er Jahren erlaubt PDM das Standardisieren, Speichern, Verwalten und Verknüpfen von CAD Daten. Zusätzlich können verschiedene Revisionsstände verwaltet werden sowie zu jedem Bauteil die Bill of Materials (BOM) abgelegt werden. **Saaksvuori.2005**

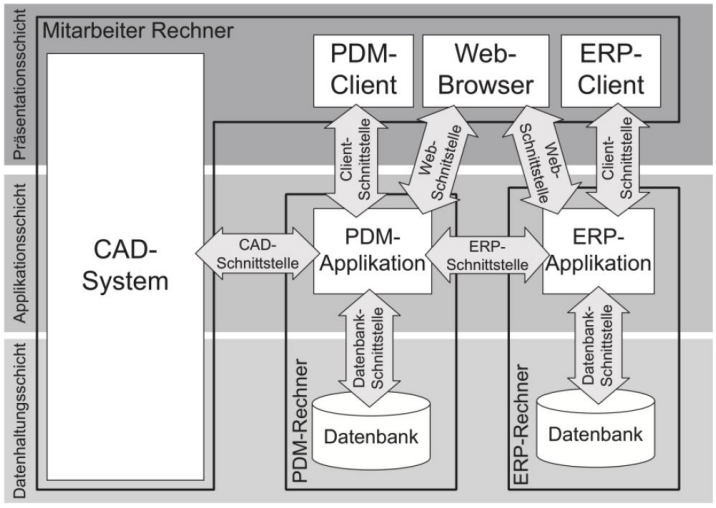


Abbildung 1: Datenverwaltung mit Standardsoftware im drei Schichtenmodell nach **Arnold.2011**

Die digitalisierten Datenbanken erlauben auch das automatisierte Verwenden von Schnittstellen, zum Beispiel zum Enterprise Ressource Planning (ERP). Arnold et al. **Arnold.2011** beschreiben Softwaresysteme, welche CAD Daten sichern und zur Verfügung stellen in drei Schichten (vgl. Abbildung 1):

1. Darstellung dem Anwender gegenüber (Präsentationsschicht)

2. Verarbeitungslogik (Applikationsschicht)
3. Datenhaltung bzw. Zugriff auf die Daten (Datenhaltungsschicht)

Das Product Lifecycle Management (PLM) organisiert IT-gestützt alle Informationen hinsichtlich eines Produkts und dessen Entstehungsprozess über den vollständigen Lebenszyklus des Produkts hinweg **Arnold.2011**. Heute gliedert sich PDM in ein ganzheitliches PLM ein.

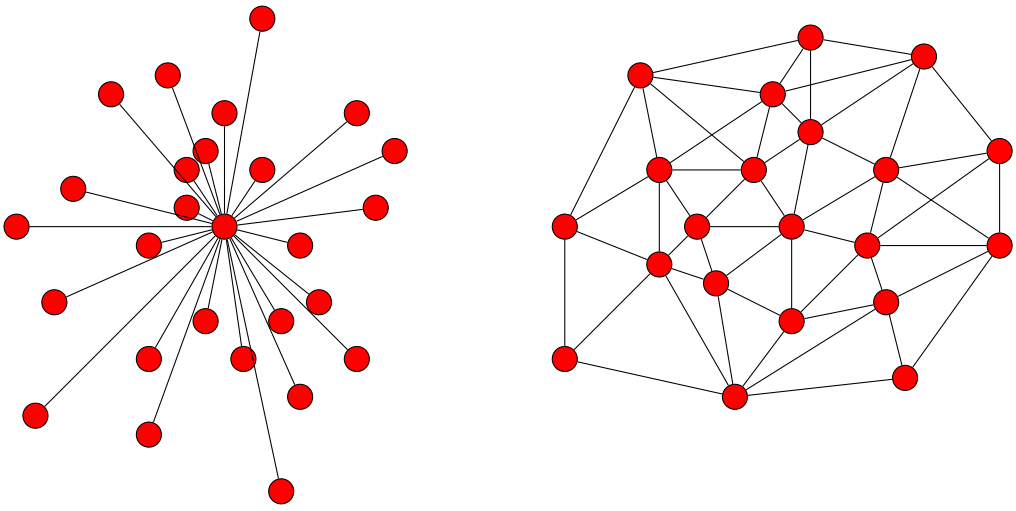
Diese Art der Datenablage impliziert einen zentralen Speicher, auf dem die Daten abgelegt werden. In der Regel ist dies ein vom Anbieter der verwendeten CAD Software angebotener Cloud Service. Daraus resultiert eine verstärkte Abhängigkeit des Anwenders vom Softwareanbieter, da er nun nicht mehr nur dessen Software verwendet, sondern auch auf die Infrastruktur des Anbieters angewiesen ist.

Die Vorteile zentraler Dateiverwaltung liegen auf der Hand. Sind alle Daten an einem zentralen Ort gespeichert, sind sie immer schnell zu finden. Bei entsprechender Verfügbarkeit des Servers sind die Daten immer und von überall aus verfügbar. Die Geschwindigkeit zentral ausgerichteter Netzwerkarchitekturen übersteigt die von verteilten Systemen deutlich, sofern entsprechend dimensionierte Server und Netzwerkanbindung gegeben sind. Darüber hinaus ist es deutlich leichter, Daten auf einem zentralen Server vor ungewolltem Zugriff zu schützen, als wenn diese verteilt auf mehreren Systemen liegen. Das Problem ist jedoch häufig die Integrität solcher Daten. In Kapitel 2.2 wird der Aspekt der Datenintegrität eingehender untersucht. Das Hauptproblem der Datenintegrität bei zentral abgelegten Dateien ist das erforderliche Vertrauen in Dritte. Die Betreiber der Datenbanken, im aktuellen Beispiel Datenbanken für CAD Daten, sind häufig Dritte. Dieser Dritte muss dafür sorgen, dass die Daten nicht oder nur von dazu berechtigten Menschen geändert werden, ohne dass die Einhaltung dieser Vorgaben von außen überprüfbar ist. Die Integrität der Daten ist also nur insoweit vorhanden, wie auch Vertrauen in den zuständigen Dritten vorhanden ist. Eine mögliche Lösung für diese Problemstellung wird in Kapitel 2.2 vorgestellt.

2.2. Blockchain Technologie

Der Begriff Blockchain ist bisher nicht eindeutig definiert **Mattila.2016**. In der Literatur finden sich unterschiedliche Definitionen für diesen Begriff **Schlatt.2016**. Im weiteren Verlauf dieses Buches wird der Begriff Blockchain im Sinne der Definition von Dannen verwendet. Demzufolge ist eine Blockchain ein verteiltes peer-to-peer Netzwerk, welches Kryptographie anwendet, um Programme auszuführen, Daten zu speichern und digitale Währungen auszutauschen **Dannen.2017**. Laut Bengel sind in einem peer-to-peer Netzwerk alle Teilnehmer gleichberechtigt und können gleichermaßen eigene Dienste anbieten sowie die Dienste anderer nutzen**Bengel.2014**.

Das verteilte Blockchainnetzwerk besteht aus vielen Netzknoten (Rechnern), die zwar miteinander verbunden sind, aber unabhängig voneinander sind. Das bedeutet, dass der Ausfall eines Knotens die übrigen Knoten nicht beeinflusst. Das wird dadurch erreicht, dass der aktuelle Zustand der Blockchain redundant auf jedem einzelnen Knoten gespeichert ist.



Zentralisiertes Netzwerk Verteiltes Netzwerk

Abbildung 2: Gegenüberstellung zentralisierter und verteilter Netzwerkarchitektur

Der unterschiedliche Aufbau der in Kapitel 2.1 beschriebenen Netzwerkarchitektur sowie der von der bei der Blockchain Technologie zum Einsatz kommenden Architektur ist in Abbildung 2 veranschaulicht. In einem verteilten Netzwerk stellen alle Teilnehmer ihre jeweiligen Ressourcen dem System zur Verfügung und ein zentraler Server ist nicht mehr erforderlich. Daraus entsteht nach Tannenbaum et al. der Vorteil von hö-

herer Rechenleistung, Kostensenkungen, einem Zuwachs an Zuverlässigkeit durch aufgebaute Redundanz sowie die Fähigkeit zu natürlichem Wachstum **Tanenbaum.2008**. Allerdings identifizieren Tannenbaum et al. auch diverse Nachteile, die die Architektur verteilter Systeme mit sich bringt. So ist der Aufwand für die Koordinierung eines solchen Netzwerkes und die Kommunikation darin deutlich höher als in zentralisierten Netzwerken, was eine erhöhte Programmkomplexität erfordert. Außerdem ist die Performance verteilter Netzwerke stark abhängig von der Netzwerkanbindung der einzelnen Teilnehmer.**Tanenbaum.2008**.

Die verteilte Netzwerkarchitektur ist die Antwort der Blockchain Technologie auf das in Kapitel 2.2 bereits angesprochene Integritätsproblem. Boritz et al. beschreiben in **Boritz.2005** Integrität als aus drei Hauptkomponenten bestehend:

Datenintegrität: Die im System verwendeten und gepflegten Daten sind vollständig, korrekt und frei von Widersprüchen.

Verhaltensintegrität: Das System verhält sich wie beabsichtigt und weißt keine logischen Fehler auf.

Sicherheit: Das System ist in der Lage, den Zugriff auf Daten und Funktionen auf autorisierte Anwender zu beschränken.

Diese Hauptkomponenten werden von verschiedenen Blockchainlösungen unterschiedlich umgesetzt. Das Hauptziel der Blockchain ist die Integrität in einem verteilten System zu erzielen und zu erhalten **Drescher.2017**. Dieses verteilte System verfügt über keine Zentrale Instanz, sondern besteht ausschließlich aus gleichberechtigten Knoten, die in direktem Austausch miteinander stehen. Im Folgenden wird das Prinzip Blockchain anschaulich am Beispiel von Bitcoin erläutert. Darauf folgend werden die zwei Open Source Blockchain Lösungen Hyperledger Fabric und Ethereum vorgestellt.

2.2.1. Bitcoin

Die Idee zu Bitcoin wurde 2008 unter dem Pseudonym Satoshi Nakamoto in **Nakamoto.2009** vorgestellt. Die ursprüngliche Motivation zur Entwicklung von Blockchain war die Umsetzung von Transaktionen einer digitaler Währungen, dem Bitcoin. Später wurde die Bitcoin Blockchain jedoch für die Übertragung und Verifizierung vieler anderer Informationsformen verwendet **CampbellVerduyn.2018**. Die Blockchain fungiert hierbei als Transaktionsregister, welches lokal auf allen Knoten gespeichert ist. Dieses Transaktionsregister muss stets den aktuellen, korrekten Zustand der Blockchain widerspiegeln.

Der Zustand der Blockchain ergibt sich aus den Kontoständen der Teilnehmer und den Transaktionen zwischen ihnen. Konkret enthält der Speicher jedes Knotens der Blockchain eine komplette Kopie der Blockchain (Transaktionsregister), einen Cachespeicher mit noch nicht verwendeten Transaktionsoutputs sowie eine Datenbank mit geprüften, aber noch nicht in der Blockchain hinterlegten Transaktionen **Franco.2015**.

Die im Folgenden beschriebenen Abläufe sind in Abbildung 3 grafisch dargestellt. Um in der Bitcoin Blockchain eine Transaktion tätigen zu können, benötigt man den Zugang zu einem Konto bzw. einer Adresse innerhalb der Blockchain. Die Authentifizierung dazu funktioniert über Public-Key-Kryptographie. Dabei wird über einen mathematischen Algorithmus ein Schlüssel-Schloss-Paar generiert. Das Schloss, der sogenannte Public-Key, liegt in öffentlich in der Blockchain. Den Schlüssel dazu, den sogenannte Private-Key, behält der Kontobesitzer. Will er eine Transaktion tätigen, muss er seine entsprechende Nachricht an die Blockchain mit seinem Private-Key signieren. Der Nachrichtentransfer wird asymetrisch verschlüsselt, sodass die Nachricht nicht unbemerkt verändert werden kann **Stallings.2003**.

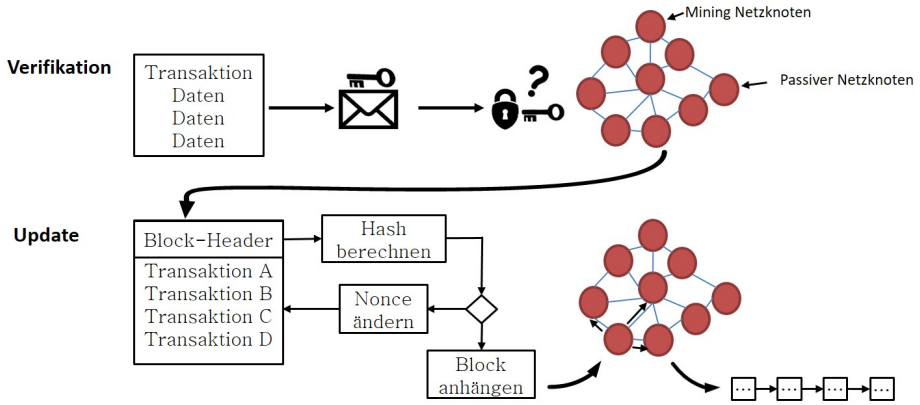


Abbildung 3: Transaktionsablauf in der Blockchain in Anlehnung an Schlatt et al. **Schlatt.2016**

Zur Absicherung der Daten in der Blockchain wird eine kryptographische Hashfunktion verwendet. Jede Veränderung der Eingangsdaten dieser Hashfunktion führt zu einer Veränderung des berechneten Hashwertes **Schlatt.2016**. Von einem Hashwert ausgehend ist es nicht mit vertretbarem Aufwand möglich den Eingangswert zu bestimmen **Franco.2015** oder einen weiteren Eingangswert zu finden, der diesen Hashwert ergibt **Schafer.2003**.

Versendet nun der Inhaber einer Adresse in der Bitcoin Blockchain eine signierte Nach-

richt mit einer oder mehreren Transaktionen, die er ausführen möchte an das gesamte Netzwerk der Blockchain **Antonopoulos.2015**. Dort landet die Nachricht erst einmal im Cachespeicher der Knoten. Jeder Knoten überprüft Signatur und Inhalt der Nachricht und leitet sie bei Bestehen der Prüfung an die übrigen Knoten weiter, die die Transaktion ebenfalls überprüfen. Die geprüfte Nachricht wird dann in die Datenbank mit den unbestätigten Transaktionen aufgenommen. Ein Mining-Knoten kann nun mehrere unbestätigte Transaktionen in einem Block zusammenfassen, um diesen der Blockchain (also dem Transaktionsregister) hinzuzufügen.

Wie in Abbildung 3 zu sehen, gibt es Mining-Knoten und Passive Knoten. Nur Mining-Knoten können Blöcke zur Blockchain hinzufügen. Um die Datenintegrität der Blöcke innerhalb der Blockchain zu gewährleisten, gibt es einen Konsensmechanismus. Dieser beruht auf einem sogenannten Proof of Work (PoW), dem Lösen eines rechenintensiven mathematischen Problems. Dazu wird ein Block aus Transaktionen und einem Block-Header zusammengesetzt (vgl. Abbildung 3). Der Block-Header umfasst eine Referenz zum vorherigen Block, einen Zielwert, einen Zeitstempel, die Wurzel eines Merkle-Baumes (eine Zusammenfassung aller enthaltenen Transaktionen in Hashform) sowie eine veränderbare Nonce **Antonopoulos.2015**. Der Mining-Knoten muss nun die Nonce variieren, bis der Hash-Wert des Block-Headers den Zielwert unterschreitet **Zohar.2015**. Der Mining-Knoten, welcher das Problem als erstes löst, hängt den Block seiner Blockchain an und teilt ihn mit den anderen Knoten, welche das berechnete Ergebnis überprüfen und dann ebenfalls anhängen. Lösen mehrere Knoten das Problem zeitgleich, entstehen mehrere Abzweigungen der Blockchain. Letztlich wird ein Knoten jedoch seine Blockchain verwerfen, sobald er von einer längeren erfährt und diese übernehmen **Zohar.2015**. Eine Transaktion gilt dann als durchgeführt, wenn sie der Blockchain angehängt wurde **Bohme.2015**.

Dadurch, dass der Hash-Wert jedes Blockes eine Referenz auf den vorangegangenen Block enthält, der wiederum eine Referenz auf den ihm vorangegangenen Block enthält, sind die Blöcke derart miteinander verkettet, dass bei einer Manipulation eines früheren Blockes die Hash-Werte aller folgenden Blöcke falsch werden. Dies würde von den übrigen Knoten bemerkt und die manipulierte Blockchain verworfen. Diese Unveränderbarkeit der Daten innerhalb der Blockchain gewährt die Datenintegrität nach Boritz et al.. Um einen Anreiz zu schaffen, die eigenen Ressourcen für einen Mining-Knoten zur Verfügung zu stellen, erhält der Mining-Knoten, der einen weiteren Block anhängt, eine

Belohnung in Form von Bitcoins. Da die Bitcoin Blockchain ein Open-Source Projekt ist, kann der Code von jedem Interessierten eingesehen und geprüft werden, um die Verhaltensintegrität des Systems festzustellen. Der in Kapitel 2.2 beschriebene Sicherheitsaspekt der Integrität im Sinne von Boritz et al. **Boritz.2005** ist in der Bitcoin Blockchain nicht gegeben, da jeder die Transaktionen innerhalb der Blockchain einsehen kann. Allerdings herrscht dennoch Anonymität, solange die Adressinhaber nicht bekanntgeben, welche Adresse welcher Person zuzuordnen ist.

2.2.2. Hyperledger Fabric

Hyperledger ist ein 2015 von der Linux-Foundation ins Leben gerufenes Open Source Ledger Framework, welches jedoch ohne eine Kryptowährung auskommt **Laurence.2018**. Hyperledger ist ein Sammelprojekt, welches die fünf Blockchain Frameworks Sawtooth, Iroha, Fabric, Burrow und Indy sowie einige Tools zum Umgang mit diesen Frameworks beinhaltet **HyperledgerWebpage**. Ziel des Hyperledger Projekts ist es, Lösungen für den Einsatz in großen Unternehmen zu konzipieren **Laurence.2018**. Im Folgenden wird aus dieser Liste nur Hyperledger Fabric beschrieben.

Wie auch bei Bitcoin sind alle eingetragenen Datensätze unveränderbar. Im Gegensatz zur Bitcoin Blockchain sind in Hyperledger Fabric getätigte Transaktionen jedoch privat und vertraulich **Laurence.2018**. An einer Fabric Blockchain können sich also nur autorisierte Personen beteiligen. Die getätigten Transaktionen können nicht mit Teilnehmern in Verbindung gebracht werden und der Inhalt jeder Transaktion wird verschlüsselt, sodass nur dafür vorgesehene Teilnehmer ihren Inhalt lesen können **Laurence.2018**. Hyperledger Fabric soll möglichst zugänglich sein, unabhängig vom technischen Erfahrungsgrad. Aktuell ist die Anwendung aber immernoch technisch sehr erfahrenen Benutzern vorbehalten **Laurence.2018**.

Nachdem Ethereum sogenannte Smart Contracts eingeführt hat, hat Hyperledger eine eigene Version entwickelt. Smart Contracts erlauben es, Vereinbarungen zwischen Parteien auf eine vertrauenswürdige Art und Weise zu automatisieren. Hyperledger befindet sich aktuell noch in einer frühen Entwicklungsphase und hat etwa zwei Jahre Rückstand auf Ethereum **Laurence.2018**. Da die Arbeit mit Hyperledger Fabric die Contrainersoftware Docker voraussetzt, ist ein Arbeiten mit Hyperledger Fabric unter Windows erst ab Version 10 ohne Umstände möglich **LeHors.2017**.

2.2.3. Ethereum

Die Idee von Ethereum wurde 2013 das erste mal in **Buterin.2013** von Buterin beschrieben. Ziel war es, die Blockchain nicht nur zum Währungsaustausch zu verwenden, sondern ihr Anwendungsgebiet auf Wirtschaft und Verwaltung auszudehnen **Laurence.2018**. Dies soll durch die Einführung von Smart Contracts erreicht werden **Dannen.2017**. Solche Smart Contracts sind Verträge, die als Programmcode geschrieben werden. Deshalb werden sie gemeinhin oft als Dezentralisierte Apps (DApp) bezeichnet.

Aufbau und Funktionsweise von Ethereum ähneln stark der von Bitcoin, denn Ethereum baut auf Bitcoin auf. Ähnlich wie Bitcoin gibt es auch in Ethereum eine Währung, den Ether (ETH) **Dannen.2017**. Der Ether soll jedoch mehr sein, als nur eine Währung. Er wird auch als commodity (dt. Rohstoff) bezeichnet. Die Ausführung von Smart Contracts benötigt diesen Rohstoff, vergleichbar mit Treibstoff **Dannen.2017**. In Bitcoin kann lediglich eine Währung von einem Ort zum Anderen transferiert werden. In Ethereum kann Ether verwendet werden, um Programme jetzt oder zeitverzögert in der Zukunft auszuführen, die dann beispielsweise Ether von einem Ort zum Anderen transferieren. Wie in Bitcoin auch werden Miner in der eigenen Währung bezahlt, um neue Blöcke mit Transaktionen zu generieren. Aktuell verwendet Ethereum ähnlich wie Bitcoin den PoW als Konsensmechanismus. Geplant ist jedoch, diesen in absehbarer Zukunft durch einen Proof of Stake (PoS) zu ersetzen**Dannen.2017**. Der PoS verwendet eine Zufallszahl und den sogenannten Stake, der sich aus weiteren Faktoren zusammensetzt, um zu bestimmen, welcher Miner den nächsten Block anhängen darf **Buterin.2013b**. Im Gegensatz zum PoW ist der Zeit- und Rechenaufwand dabei deutlich geringer. Im Gegensatz zu Hyperledger Fabric gibt es in Ethereum keine private oder permissioned Blockchain, es ist lediglich möglich, eigene Netzwerke in LAN-Netzwerken zu erstellen und das gesamte LAN-Netzwerk nach außen abzuschirmen. Außerdem ist Ethereum auch als Windows Anwendung verfügbar.

Für Smart Contracts wird in Ethereum die eigene Programmiersprache Solidity verwendet **Dannen.2017**. Die aktuellste verfügbare Version zum Zeitpunkt des Erscheinens dieser Arbeit ist Version 0.4.24 **SolidityRepo**. Daraus geht hervor, das Solidity noch weit entfernt davon ist, den gewünschten Funktionsumfang vollständig zu bieten. Der Programmierer muss bei Verwendung von Solidity zum aktuellen Zeitpunkt also noch mit einigen Einschränkungen rechnen. Smart Contracts erhalten, wenn sie in die Block-

chain hochgeladen wurden, eine eigene Adresse und werden vom Ethereum Netzwerk ausgeführt, sofern sie mit ausreichend Ether versorgt sind. Um einen Smart Contract auszuführen, müssen ausreichend Rohstoffe bereitgestellt werden, um den Programmcode auszuführen und seine Daten in der Blockchain abzulegen **Dannen.2017**. Dann werden alle Daten im nächsten Block und damit redundant auf jedem Knoten der Blockchain abgelegt. Die für Javascript bereitgestellte Web3.js Bibliothek ermöglicht es dem Programmierer Programme zu schreiben, die mit einem in Solidity geschriebenen Smart Contract in der Ethereum Blockchain interagieren. Wie genau Daten in einem Smart Contract abgelegt werden können, wird in Kapitel **??** genauer betrachtet.

3. Section I

Lalala

4. Deskriptive Phase II

A. Appendix

Erklärung

Hiermit erkläre ich gemäß der Rahmenprüfungsordnung der Hochschule München, dass ich die vorliegende Arbeit mit dem Titel "Implementierung einer Blockchain-Anwendung zur Steigerung der Integrität von CAD-Metadaten" selbstständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benutzt sowie wörtliche und sinngemäßge Zitate als solche gekennzeichnet habe.

München, den _____ Unterschrift _____