Abstract

With the introduction of blockchain technology a new type of platform has been created that enables the development of decentralized applications based on smart contracts. Smart contracts are computer programs that get deployed on the blockchain and embody encoded agreements, which get executed by nodes within the blockchain network. DApps describe decentralized applications that are built on top of blockchain technologies, such as smart contracts, which in some cases can substitute centralized servers. However, the current adoption of DApps primarily takes place in financially related categories and appears to be insignificant with regards to the expectations of the potential of blockchain technology. As usability can impact the adoption of applications, this thesis examines the usability challenges DApps face. More specifically, insights are gathered from a technical perspective and from the perspectives of both developers and users of DApps through cognitive walkthroughs and semi-structured interviews, to provide an overview of existing usability challenges. Furthermore, recommendations and requirements to overcome these identified challenges get proposed. With technological advancements evolving at a significant pace, standards, as well as novel onboarding services, that require less effort for the user to interact with DApps, are beginning to emerge and challenges, such as the scalability of transactions, get alleviated. However, interviews with DApp developers reveal, that the technical emphasis of the industry gets reflected in the terminology that is used in DApps, which can hinder the learning process of users unfamiliar with blockchain technology. Moreover, it gets pointed out, that price volatility, financial incentivisation and speculation can pose challenges to the usability of DApps. To promote usability, it is suggested to focus more on the individual user and to enable interactions that users might be more familiar with by hiding blockchain related mechanisms.

Contents

List of Figures

List of Tables

List of Abbreviations

ABI	Application Binary Interface
ACM	ACM Digital Library (2018)
AML	Anti Money Laundering
CL	Cognitive Load
CW	Cognitive Walkthrough
DAG	Directed acyclic graph
DApps	Decentralized Applications
DeFi	Decentralized Finance
DLT	Distributed Ledger Technology
DpoS	Delegated Proof of Stake
ENS	Ethereum Name Service
EVM	Ethereum Virtual Machine
ERC	Ethereum Request for Comment
HCI	Human Computer Interaction
ICO	Initial Coin Offering
IEEE	IEEE Xplore Digital Library (2018)
IoT	Internet Of Things
ISO	International Organization for Standardization
KYC	Know Your Customer
NAT	Network Address Translation
NFT	Non-Fungible Token
OTC	Over-the-counter
PoS	Proof of Stake
PoW	Proof of Work
SDK	Software Development Kit
UX	User Experience

1 Introduction

DApps (Johnston et al. 2015, pp. 2,3) describe decentralized applications, which are open source software systems that are based on blockchain technology and utilize a consensus- and token generation mechanism (describing the issuance of virtual assets, which are necessary for the usage of the DApp) (Johnston et al. 2015, p. 3). The business logic of a DApp can be implemented by smart contracts, which represent computer code that gets executed in a virtual machine through a distributed set of nodes (computers) in a network in a decentralized manner (Wessling and Gruhn 2018, p. 45). To this day[1], more than 2287 DApps have been launched using the smart contract platform Ethereum, making it the most prominent platform for the creation of dapps. Decentralized applications are being published at an accelerating pace, with more than 178 publicly available DApps being released in the month of December 2018, with most DApps being developed in the context of gaming, storing and exchanging digital assets (Wu 2019, p. 3,4).

As of now, the amount of daily end users utilizing decentralized services based on blockchain technology remains relatively small in contrast to users of non-decentralized services (State of the DApps 2018). Since the experience with a given system or product can have a significant impact on its acceptance and adoption (Zhou 2012, pp. 27-37), investigating the usability of existing decentralized applications could provide reasons for this phenomenon. Furthermore, while previous blockchain-related research has extensively covered technical aspects (Porru et al. 2017), vulnerabilities (Delmolino et al. 2016) and limitations of smart contracts (Luu et. al 2016), little research has been done to examine the specific usability and user experience of decentralized applications. No overview of prevalent practices has been outlined, no respective standards or guidelines have been defined yet in academic literature and the specific assessment of DApps in terms of usability or user experience hasn't been the focus of any previous study[2].

[1] As of Thursday, 14th March 2019.

[2] As a search in the literature databases ACM and IEEE with the combinations of terms „blockchain", „Ethereum", „dapp", „decentralized application" and „user experience", „usability", „ux" shows.

The disparity of expectations regarding the potential of blockchain technology (Deloitte 2018, p. 19) and actual user count of decentralized applications poses questions targeting the usability: *What are the challenges in enabling an appropriate usability for decentralized applications – and what requirements can be formulated that aim at overcoming those challenges?*

The main goal of this thesis is to answer both questions by gathering insights from two different perspectives: A technological perspective, which investigates the technical layer of DApps with its constraints and characteristics and the perspective of the agents who create and interact with the DApps: The users and developers. First, usability challenges get derived from the technical idiosyncracies of DApps. In first section of the technical perspective, the ways in which the technology behind DApps can influence the usability, is being investigated.

In order to provide a knowledge base that can be examined to derive potential usability challenges from, this study starts off with a theoretical foundation depicting the concepts and factors of usability and user experience focused design. Besides investigating what factors enable an appropriate usability, an in-depth look at the mechanics, platforms and architectural idiosyncracies of decentralized applications is being provided in order to understand the constraints and differences to non-blockchain applications.

To then investigate the usability from the perspective of an end-user, cognitive walkthroughs of DApps are subsequently being conducted, with the aim to further enrich the set of usability challenges of DApps. Additionally, aspects and qualities that are found to be helpful in promoting an appropriate usability, are mentioned. For this approach, an evaluation framework gets composed out of existing research, that helps with classifying and identifying usability challenges. Afterwards, semi-structured interviews with designers and developers working on smart contracts based DApps are being held. It is assumed, that developers can provide insights into the behavior of DApp users and related usability issues that users face. During these interviews, a selection of previously identified usability challenges (presented in chapters 3 and 4.3) get addressed by specifically asking the interview partners about their experience regarding them. Furthermore, the interview partners are being asked about their recommendations on how to provide a more appropriate usability and how to solve the discussed usability challenges.

With both the identified challenges stemming from the technical layer and the obtained empirical data from the semi-structured interviews and the cognitive walkthroughs, overarching usability related themes and issues are being identified in the next step. Intersecting areas can be identified by comparing and the clustering the gathered insights from three different perspectives. Next, the recommendations on how to overcome the identified usability challenges, that were given during the semi-structured interviews are extended by formulating recommendations based on technical advancements, as well as insights from the cognitive walkthroughs. In this step, aspects and features of the reviewed DApps that proved to contribute to an appropriate usability for a novice user, are addressed. Furthermore, requirements for establishing an appropriate usability of DApps are outlined are inferred from the set of recommendations, with respect to the existing blockchain knowledge of potential users and the category a DApp can be placed in. Afterwards, the findings are being summarized, followed by an outlook on future research directions.

This thesis contributes to the knowledge base by providing an overview of the current state of the usability of decentralized applications and by proposing and discussing approaches on how to tackle the identified usability challenges with the overall goal of improving the usability for the end-user of DApps. As research in exploring the user experience and usability of DApps can mostly be found in non-scientific, open-source and collaborative documentations, this thesis can be useful for Ethereum DApp developers by offering aggregated related insights in a comprehensive and scientifically backed way. Moreover, the provided recommendations can guide the development of DApps up to an architectural level by adding and emphasizing the perspective of the user. Considering that a high usability and delightful user experiences can also help with software adoption, the Ethereum DApp developer community, which is driven by a constant advancement of its ecosystem through decentralized collaboration, can be positively affected by this work.

Given the complexity of the dynamic blockchain ecosystem with new protocols and smart contract enabling platforms launching at an increasing rate, this thesis solely aims at investigating the DApps built on the main and public blockchain of Ethereum. Ethereum, which launched in 2015, is a smart contract enabling platform with more than 54124 smart contracts deployed to its blockchain (Etherscan 2018) and features a detailed documentation. Furthermore, in the following chapters, the term adoption is used synonymously with the term acceptance and the word recommendation is used synonymously with the word requirement.

In the existing body of related research that contributes to the knowledge base of this study, Gruhn and Wessling find out that, even though blockchain technology and smart contracts are represented in the business logic of a software system, various architectural patterns of decentralized applications exist, which each offer a certain grade of trade-off between convenience and trust and that can have a significant influence on the user experience (Wessling and Gruhn 2018, pp. 45,46). Chapter 4, which focusses on reviewing the usability of selected DApps using cognitive walkthroughs, builds on the findings Wessling and Gruhn gathered and seeks to expand on these insights.

Eskandari et al. (2018, pp. 1-10) investigated the usability challenges of key management of Bitcoin users and found out that the usage of private key based authentication for client software dealing with cryptocurrencies, which is hardly used by non-experts, can create various usability challenges, especially for novice users. In the context of this thesis, investigating the influence the form of authentication (as part of chapter 4) can have on the usability, can be seen as an extension to the study.

Khairuddin and Sas (2017, pp. 6499-6510) investigated the concept of trust in dealing with bitcoin transaction. Trust is a factor that is commonly measured using questionnaires in the context of user experience evaluation (Sauro 2015, pp. 68-86). Khairuddin and Sas found out, that while decentralization, unregulation, as well as transparent and low cost transactions improve the sense of trust, the risk of insecure transactions and dishonest traders pose the most significant challenge in terms of trust. Therefore, a further step is to investigate the role of trust within the usage of DApps, which are based on the transactions of cryptocurrencies.

2 Theoretical Foundation

This section is divided in separate chapters and provides detailed background information about the concepts of both user experience and usability, as well as about the fundamental mechanisms that enable smart contract based applications to function. Given the complexity of blockchain related topics, this section specifically focusses on subjects that are both necessary to understand the further reading and that serve as a knowledge basis to derive challenges from in section 5.

The section starts off by describing the concepts of usability and user experience, highlighting both their relation and differences. Afterwards, the functionality of blockchain technology and smart contracts is being outlined, followed by a description of its promises and use cases. Next, the technical architecture of smart contract based applications is being discussed, with an in-depth look at how smart contracts can receive external data and how processed data can be stored. Finally, this section ends with an overview of existing Ethereum-based applications.

2.1 User Experience, Usability and their Relation

With the introduction of the personal computer in the 1980s a shift in the development of computer and software systems took place: While systems had previously been primarily built for specific information technology professionals, the emerging market of personal computers required a deeper understanding of how users interact with computer and software systems. With that, the field *Human Computer Interaction* (HCI) began to evolve as a new research area of computer science (O'Regan 2018, pp. 147,152). In the beginning, HCI research mostly focused on examining instrumental qualities of products to assess how "usable" given products are (Hassenzahl et al., 2006, p. 92). Evaluation methods, such as usability tests, were introduced to measure the level of achievement of specific goals and tasks. Soon researchers and practitioners became aware that non-pragmatic qualities, such as aesthetic or emotional qualities can shape the usage of a product as well and that the functional focus of usability evaluation should be extended by also taking into account subjective aspects and product experiences. With that, the field of User Experience was formed with the aim of enriching the existing area of HCI research by adding non-functional qualities as subjects to investigate (Hassenzahl et al., 2006, pp. 92-93).

2.1.1 Definitions of Usability

Usability is considered one of the most widely disseminated concepts of Human Computer Interaction (HCI), which features a higher agreement on its definition in academia than the concept of user experience (Thielsch et al. 2009, p. 241).
Within the 9241 standard, the *International Organisation for Standardization* (ISO) defines usability in part 11 as "the effectiveness, efficiency and satisfaction with which specified users achieve specified goals in a specified context of use." Furthermore, the ISO elaborates on its definition by explaining effectiveness as "the accuracy and completeness with which users achieve specified goals" and efficiency as "the resources expended in relation to the accuracy to the use of the product". Satisfaction is being defined as the "freedom from discomfort, and positive attitude to the use of the product" and the "specified context of use" as the "characteristics of the users, tasks and the organizational and physical environments" (Jokela et al., 2003, p. 54).

In the part 110 of the ISO 9241 standard (2006), general ergonomic design principles are being defined with the goal to help the "analysis, design and evaluation of interactive systems" (ISO 9241 2016, p. 4). Moreover, it is stated, that these design principles are being outlined without reference of any external circumstances and context and serve only as recommendations for the design of dialogues between users and interactive systems. The listed design principles are: "suitability for the task", "self-descriptiveness", "conformity with user expectations, "suitability for learning", controllability", "error tolerance" and "suitability for individualization" (ISO 9241 2016, pp. 7-16).

The international standard ISO/IEC 25010, which got published in 2011, describes the qualities of a software product, one of which is usability. According to this standard, usability is constituted by attributes that describe both the "effort needed for use" and the "individual assessment of such use". The ISO 25010 lists the following attributes as relevant to usability: "User error protection", "accessibility", "appropriateness recognizability", "user interface aesthetics", "learnability", "operability" and "usability compliance" (ISO 25010).

According to Nielsen (1993), the concept of usability describes how well users can use the functionality provided by a given system ("utility") and it thereby encompasses all "aspects of a system with which a human might interact" (Nielsen 1993, p. 25). Moreover, Nielsen argues that usability depicts multiple properties of an interface that can be evaluated using usability assessment methods, such as "user testing" (Nielsen 1993, p. 165 ff.). Nielsen differentiates between five attributes that constitute his definition of usability: "Learnability", "efficiency", "memorability", "errors" and "satisfaction" (Nielsen 1993, p. 26). In order to evaluate a "system's overall usability", Nielsen states that it is necessary to measure each attribute, with "learnability" being the easiest and "satisfaction" being the hardest attribute to measure (Nielsen, 1993, p. 27).

Alonso-Ríos et al. (2009, p. 3 ff.) argue, that existing definitions of usability are brief, informal and imprecise and that neither reasearchers nor standards have come to one definition on usability that is collectively agreed on. In an attempt to unify existing definitions, they propose a taxonomy of attributes defining usability that they created by comparing and analysing various existing definitions of usability. According to their model of usability, usability consists of the attributes "knowability", "operability", "efficiency", "robustness", "safety" and "subjective satisfaction".

Cognitive Load (CL) is a concept that got first introduced in the field of cognitive psychology and refers to the observation, that the ease of completing a task depends on the required capacity of the user's working memory (Schmutz et al. 2009, pp. 1,2). As "learnability" and "effort needed for use" partly constitute usability (ISO 25010), cognitive load can thus impact the usability. Therefore, the process of designing user interfaces factors in the "capacity limitation" of the different users, which means, that the visual layout, the presentation of the information and the way the user interacts with the system should to be designed in a way that reduces the user's short term memory load (Schmutz et al. 2009, pp. 1,2). Paas et al. (2003, pp. 1-3) differentiate between three different sources of cognitive load: "Intrinsic cognitive load", "extraneous cognitive load" and "germane cognitive load". "Intrinsic cognitive load" refers to the inherent complexity of the context of a given task, which, according to Paas et al., can't be reduced by instructional design. "Extraneous cognitive load" describes the cognitive effort that is induced through an inefficient presentation of the task and the topic, while "germane cognitive load" denotes the cognitive work required for learning processes. In order to optimize a given user interface in terms of cognitive load, it is suggested to minimize extraneous load through an appropriate presentation of information and task instructions and to increase germane load by stimulating learning-related processes (Schmutz et al. 2009, p. 2). Mandel (1997, pp. 63-71) proposes nine design principles to reduce the user's cognitive load, which include using "real-world metaphors", "providing defaults" and allowing the user to "undo" and "redo" activities and not forcing the user to recall information he has to enter.

Various methods exist to assess the usability of a given system (Hollingsed and Novick 2007, pp. 1-5). According to Nielsen (1994), four general ways exist, in which a usability evaluation can be conducted: A user interface can be assessed automatically (using a dedicated program that measures the fulfillment of a set of usability requirements), empirically (with real users who test the interface), formally (using exact specifications to calculate deviations) or informally (relying on "rules of thumb" and the experience of the reviewer). One of the existing usability assessment methods is the cognitive walkthrough method, which will be described in chapter 4.1.

2.1.2 Definitions of User Experience

Despite growing interest in both industry and academia and its exploration and acceptance in the field of HCI, previous research indicates that no clear consensus of the definition of the term "user experience" and its scope exists (Law et al. 2009, p. 719) and that it is being critiqued as being vague and indistinct (Hassenzahl et al. 2006, p. 91). Law et al. (2009) see possible reasons for the variety of definitions in the diversity of the underlying concepts, such as the emotional, aesthetic or hedonic variables involved in an experience. Moreover, Law et al. (2009, p. 719) argue that units to be investigated can be too flexible and that research in user experience is scattered and conducted with different emphases and theoretical perspectives. Despite its heterogeneity of definitions, user experience is generally understood as a dynamic, context-dependent and subjective concept, that is new in the context of HCI research (Law et al. 2009 p. 727). Furthermore, Law et al. found out, that the understanding of user experience among practitioners and researchers can significantly vary based on the individual working experience (p. 727).

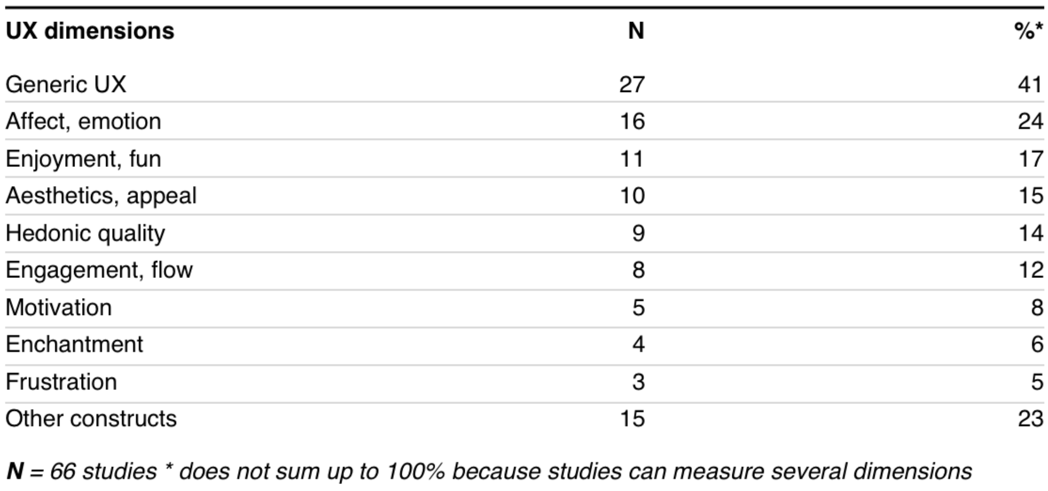
The international organization for standardization defines user experience within ISO 9241-210:2010 as a "person's perceptions and responses resulting from the use and/or anticipated use of a product, system or service". Furthermore, it is being mentioned, that "user experience includes all the users' emotions, beliefs, preferences, perceptions, physical and psychological responses, behaviours and accomplishments that occur before, during and after use".

According to Bevan (2009, p. 3), the ISO 9241-210 definition is ambivalent in terms of describing usability as inherent to user experience. As the definition includes "all behavior" of the user, it therefore also includes the user's "effectiveness and efficiency". Bevan points out three different concepts of user experience:

- User experience as "an elaboration of the satisfaction component of usability" (Bevan 2009, p. 3)
- User experience as isolated from usability, which is solely focused on "user performance" (Bevan 2009, p. 3)
- User experience as an "umbrella term for all the user's perceptions and responses, whether measured subjectively or objectively" (Bevan 2009, p. 3)

Hassenzahl and Tractinsky (2006, p. 95) view user experience as a "consequence of a user's internal state", including his predispositions, expectations, needs and motivations, the “characteristics of the designed system” and the “context in which the interaction occurs”. Moreover, Hassenzahl and Tractinsky emphasize that the field of user experience means more than “merely preventing usablitity problems” and also comprises enabling and fostering “outstanding quality experiences” (p. 95). Consequently, they propose three dominant perspectives that each contribute to the understanding of user experience, but don’t claim to fully capture the concept of user experience: The experiential perspective, the perspective that covers emotions and affect, and the perspective that goes “beyond the instrumental”. The experiential perspective focusses on the “situatedness” and the “temporality” of the usage of technology and explains an experience as the unique combination of elements such as the product and the internal state of the user and their interplay over time. The perspective covering “emotions and affect” focusses on affect as an “andecendent, a consequence and a mediator of technology use”. Moreover, the perspective views emotions not only as the prevention of frustration, but also as the promotion of “joy, fun and pride”. The third perspective addresses the satisfaction of human needs “beyond instrumental” qualities through technological usage, such as the human need for stimulation, evocation and identification (Hassenzahl and Tractinsky 2006, pp. 92-95).

Bargas-Avila and Hornbaek (2011), who reviewed 66 empirical studies to assess the practice of user experience research, found out that in practice, emotions, enjoyment and aesthetics are the dimensions of user experience that are assessed the most, while the context of use and anticipation are the least assessed factors, as illustrated by table 1. The construct “generic UX” means that the concept of user experience hasn’t been defined by any factors that are being investigated as part of the respective empirical study. Moreover, they noticed that hedonic or non-task-orientated goals are associated with the concept of user experience, whereas pragmatic and task-orientated goals are commonly associated with usability. However, they learned that instrumental and non-instrumental goals are often “interwoven and inseperable” (Bargas-Avila and Hornbaek 2011, p. 2695) and refer to Law and van Schaik (2010, p. 314), who argue that the usage of goals and tasks between usability and user experience is “questionable”.



UX dimensions	N	%*
Generic UX	27	41
Affect, emotion	16	24
Enjoyment, fun	11	17
Aesthetics, appeal	10	15
Hedonic quality	9	14
Engagement, flow	8	12
Motivation	5	8
Enchantment	4	6
Frustration	3	5
Other constructs	15	23

***N** = 66 studies * does not sum up to 100% because studies can measure several dimensions*

Table 1: Dimensions in user experience research, adapted from Bargas-Avila and Hornbaek (2011, p. 2693)

2.2 Blockchain Technology

Blockchain technology first got introduced as the underlying technology of the cryptocurrency Bitcoin in a whitepaper published by a pseudonymous entity called "Satoshi Nakamoto" in 2008. In this whitepaper, Nakamoto stresses the dependence on third parties in current electronic transaction processing and points out the resulting additional transaction costs and uncertainties. Subsequently, Nakamoto depicts a new protocol which enables peer-to-peer transactions without intermediary third parties based on cryptographic validation (Nakamoto 2008, pp. 1-9).

A blockchain is a distributed ledger of transactional records, which is shared by nodes in a peer-to-peer network. Participating nodes own at least a partial copy of the ledger and are incentivized to verify and process transactions through a consensus mechanism, by which the validity of the blockchain and the trust in the validity of a transaction is maintained. Blockchains are being viewed as immutable, as verified payments are being grouped in so called *blocks*, which then get appended to the existing chain of transactional records. Since all blocks are connected through a hashing algorithm, each new block references its *parent block* through an unique hash, which guarantees transparency (Buterin 2014).

At its core, a blockchain records the ownership and the change of ownership of an asset. Since a blockchain changes its state with each new set of transactions, it can also be described as a state-transition-machine (Buterin 2014). Each new set of transactions, that is included in a block, represents a transition, which leads to a new state of the blockchain. Before a transaction can be send to the network, the transaction has to be assigned with a digital signature in order to make the fact, that the transaction has been sent by the emitter, publicly verifiable. Blockchains make use of public, asymmetric cryptography in order to enable digital signatures: Each user of the Bitcoin network owns a *public key* with a corresponding *private key*. The public key serves as an address send and receive Bitcoins from and is publically visible. Before emitting a transaction to the network, a hash gets created from the transactional data, which gets then encrypted with the private key of the sender and is called the *signature*. When a validating node chooses to include the transaction in a new block, it can validate the signature by decrypting the encrypted hash with the public key of the sender. Furthermore, the node creates a hash of the transactional data. When the decrypted hash matches the hash of the created data, the node can be sure that the digital signature is valid (Zheng et al. 2018, p. 356).

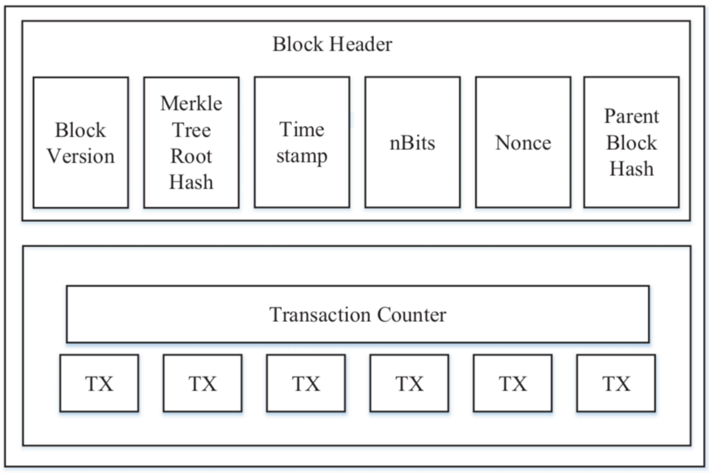


Figure **1**: Bitcoin block structure according to Zheng et al. (2017, p. 558)

A blockchain is constituted by blocks, which are interconnected through a hashing algorithm and which each include transactions. Figure 1 describes the architecture of the blockchain of Bitcoin. Each block has a block *header* and a block *body*: The transactions and a transaction counter make up the body of a block, while the *header* comprises a timestamp, the hash of its parent block, the hash value of all transaction in the *body*, a *nonce*, which is a 4-byte number, a hashing target and a *block* version. With Ethereum, a block not only stores the hash of its parent blocks, but also of its *uncle blocks*, which means the hashes of the *child blocks* of its *parent's* block are also stored (Zheng et al. 2017, p. 558). As each block, except the first, *genesis block*, references its *parent block*, all *blocks* are connected through its hashes. Moreover, since every block hash is unique and is being created using its contents, transactions that are included in an appended block are considered final and immutable. Bitcoin employs the SHA-hashing function, a deterministic, one-way hashing algorithm, which generates a 256-bit hash number and which makes sures that the input can not be derived from the output hash (Friedlmaier et al. 2018, p.3).

While Bitcoin hasn't been the first concept of a decentralized, digital currency by which transactions are validated through cryptographic, computational work (Szabo 2008), Bitcoin has been the first protocol to solve the *double spending* problem. The double spending problem describes the issue that occurs when a digital currency gets replicated and spent numerous times. Buterin (2014, p. 8) describes a scenario where one entity attempts to double spend by first sending Bitcoin to a merchant to receive an instantly delivered digital good. Upon reception of the good, the entity sends the same funds to himself, after which he tries to convince the network that the transaction to himself came first and is thus valid. Blockchain networks are protected against such attacks through the usage of a consensus mechanism. An emitted transaction doesn't get recorded on the blockchain immediately. Various types of consensus models exist to maintain the validity of the blockchain. With most consensus models, one node gets determined, which gets the chance to append its block to the blockchain upon a collective validation process: First, validating nodes, who all have the same copy of the blockchain, agree upon the integrity of the transactions and the block itself and thus come to a consensus. Only after that, the block gets appended to the blockchain and the included transactions are being confirmed. Hence, only one transaction of the attacker in the previously described scenario would have been confirmed and declared as valid (Zheng et al. 2017, pp. 559,560). As the task of validating transactions and the state of the blockchain can create a computational and financial effort, most consensus models incentivize nodes to keep track of the correct state of the blockchain.

Both Bitcoin and Ethereum currently employ the *proof-of-work* consensus model. With *Proof of work* (PoW) validating nodes are called *miners*, who try to solve a computationally intense cryptographic puzzle. The first node to successfully solve the puzzle with valid transactions gets to attach its block to the blockchain and receives the transaction fees. Furthermore, the node is allowed to include a transaction sending itself a certain amount of the respective currency (Buterin 2014). In the case of Bitcoin, the current reward is 12.5 and halves every 210.000 blocks (Anceaume et al. 2016, p. 319). Due to the economic incentive being included with a mined block, miners favor transactions with a higher mining fee (Buterin 2014, p. 27). In the case of Bitcoin and Ethereum, miners compete in creating an hash that is lower than a certain dynamic value, which is regularly adjusted to the overall computing power of the network and which dictates the difficulty and speed of the mining process. This dynamically adjusting value is computed in a way that about every 10 minutes a new block gets added. With Ethereum the average block time is 17 seconds (Zheng et al. 2018, p. 359). In order to try to win this competition, mining nodes increase the *nonce* number in a "trial-and-error"-way and create a hash of the nonce and the block until one miner achieves a hash that is lower than the predefined value (Buterin 2014, p. 7).

As computational work is linked with energy consumption, the PoW-model has been criticized for its energy expenditure and resulting environmental consequences. According to yearly projections, it is estimated that the Bitcoin network consumes as much energy as Singapur (Digiconomist 2019). To make further use of the computationally intensive calculations, some projects have modified the *PoW* mechanism to use the computing power for additional applications such as big data analysis (Matrix 2018) or mathematical research (Zheng et al. 2018, p. 359).

The proof-of-stake (PoW) model is a consensus mechanism that requires fewer energy consumption: Instead of determining the contributing node by a puzzle solving competition, the node that gets to appends the block is primarily being chosen based on its holdings of the respective cryptocurrency. However, with the idea of allowing the wealthier nodes to have a higher chance of contributing a new block and thus receiving the appertaining reward, the PoW mechanism has been criticized for having an unfair reward allocation model. This is why some projects have chosen to employ a hybrid model, which combines PoS with factors such as randomization or the timespan of holding the respective asset (Zheng et al. 2017, pp. 560-561). Ethereum, which is the focus of this study, is utilizing a modified PoW consensus mechanism called *Ethash* (Ethash, 2018), but it is planned that Ethereum will switch to a *PoS/PoW* hybrid mechanism called *Casper* in the future, by which nodes participate in a betting game in which they try to predict the block that will be included in the blockchain (Buterin and Griffith 2018, pp. 1,2). Furthermore, Ethereum's consensus mechanism is more memory-intensive than Bitcoin's consensus mechanism, as each Ethereum client needs to store a regularly generated dataset of about 1 GB in advance (Macdonald et al. 2017, p.6).

Another consensus model is *delegated proof-of-stake* (*DPoS*). While *DPoS* is similar to PoS in the sense that nodes have a higher decision power based on their holdings, representatives are being introduced, which nodes can elect and which have the task of validating blocks. Having fewer nodes that validate blocks results in smaller validation times and thus higher transaction speeds than PoS (Zheng et al. 2018, p. 360).

2.2.1 Smart Contracts

While blockchain technology based on Bitcoins protocol enables financial transactions to be processed in a decentralized way without a single, centralized entity, smart contracts broaden the range of applications of blockchain technology. Smart contracts expand the functionality of Bitcoins blockchain protocol by allowing computer code, that embodies encoded agreements, to be permanently stored in the blockchain and to be executed once certain conditions are met (Delmolino et al. 2016, p. 4). Thus, smart contracts enable the automatic transfer of ownership of goods depending on specific conditions and paylods included in the invocating transaction. Furthermore, smart contracts, once invoked, work without any downtime and interference of external third parties (Kounelis et al. 2017, p. 2).

The term *smart contract* was first coined by Nick Szabo in 1996, who defined smart contracts as a digital ”set of promises”, which are fulfilled by parties on a protocol layer (Szabo 1996). While Bitcoin allows for basic scripting and thus the creation of smart contracts on top of its protocol, it is deemed as inefficient and limited (Delmolino et al., 2016 pp. 3-4). Ethereum, introduced as “the next generation smart contract and decentralized application platform” in 2014 (Buterin 2014), has in this sense been the first platform to enable smart contructs to be written in a general purpose and Turing-complete language, such as *Solidity* (Macdonald et al. 2017, p. 6). The flexibility of smart contracts allows blockchain technology to be applied to a broader scope of applications beyond the facilitation of payments, such as financial services, supply chain applications, voting systems (Pilkington 2016, pp. 20 ff.), identity management and decentralized gambling (Friedlmaier et al. 2018, p. 6).

In order to better understand the functionality of smart contracts, a brief overview of the interaction with the Ethereum platform is being provided. The state of Ethereums blockchain relies on so called *accounts*, which each have a *nonce*, information of their respective balance. Furthermore, Ethereum differentiates between two types of accounts: *Externally owned accounts* and *contract accounts*. *Externally owned accounts* are assigned to human agents, wheareas contract accounts are controlled by their contract code that resides in the blockchain (Buterin 2014, pp. 13-14). Figure 2 by Delmolino et al. (2016, p. 3) visualizes the relations of both externally owned accounts and contract accounts.

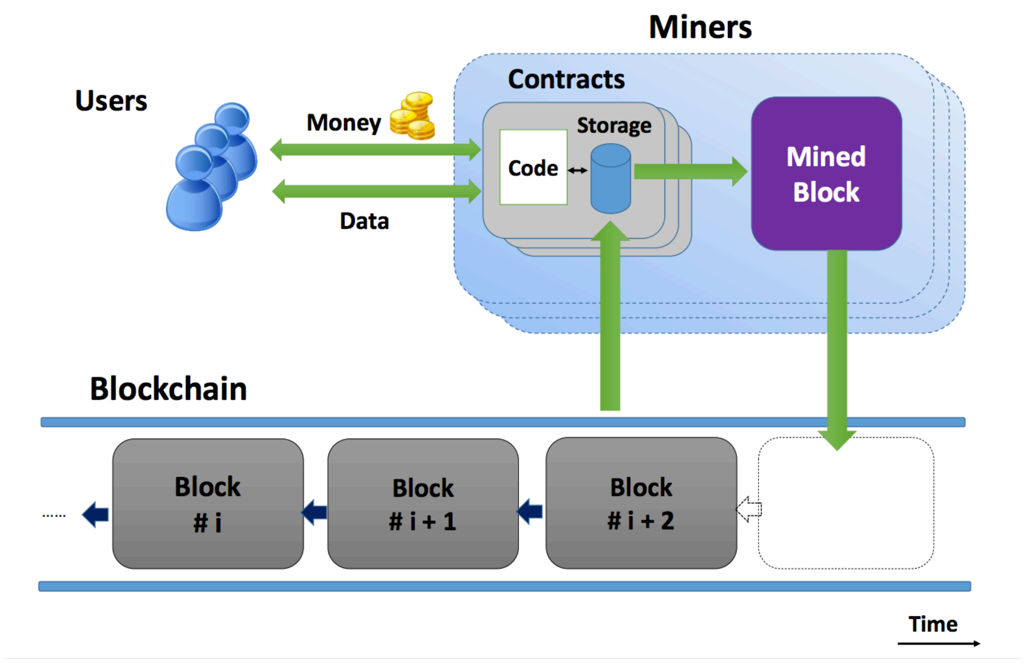


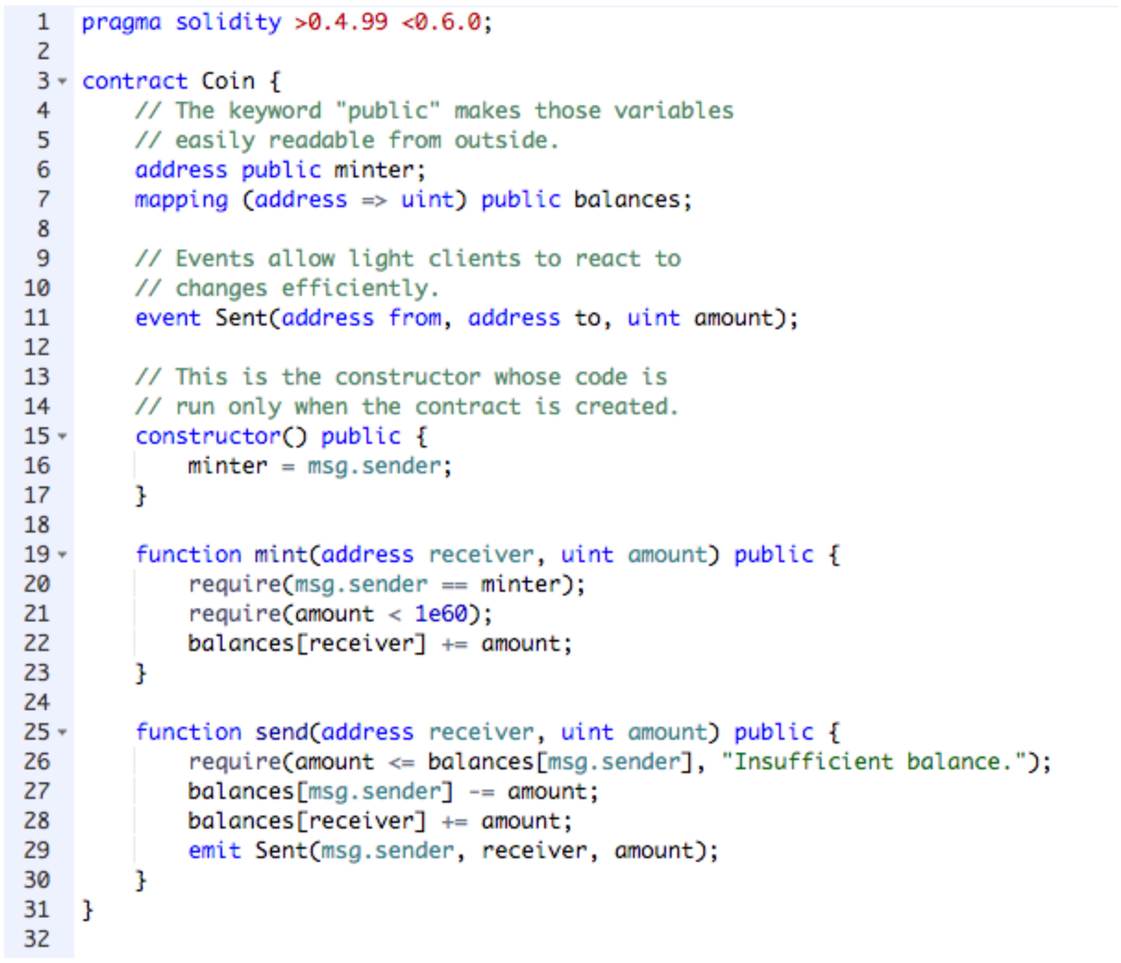
Figure 2: Ethereums smart contract platform according to Delmolino et al. (2016, p. 3)

Externally owned accounts, which represent "users", own their private keys, whereas each contract account, here represented by "contracts", don't own any private keys, but include computer code and a storage file. Both externally owned accounts and contract accounts have its own balance of *Ether*, the respective currency of the Ethereum ecosystem. *Messages*, which can be sent between externally owned accounts and contract accounts, are the equivalent to transactions in Bitcoin, as they transfer currency, but can also include data (Buterin 2014). While a smart contract can send messages back as a response, it can only be invoked by externally owned accounts or by other smart contracts, but not by itself. Ethereums runtime to execute smart contracts is its so called *Ethereum Virtual Machine* (EVM) (Buterin 2014).

Furthermore, Ethereum uses a unit to limit the computational resources used by miners to execute the contract code, which is named *gas* and included with every transaction. When deploying or executing a smart contract, the sender has to estimate the maximum amount of gas needed to execute the contract and also define his gas price he is willing to pay for each gas spent, as each computational step requires gas. Gas is similar to the mining fees included with every Bitcoin transaction in the sense that miners prefer to process transaction with a higher gas price. Thus, transactions with a higher gas price are more likely to get processed faster (Buterin 2014).

In order to create a smart contract, an externally owned accounts sends a transaction, that is signed with the sender's private key, containing Ether, which includes both the maximum gas in Ether and the senders payment value in Ether. Additionally, the transaction contains the contract code, which is compiled in EVM bytecode. As all method names and its parameters are hashed during compilation, an overview over the included methods and arguments is needed. This is why an *application binary interface* (ABI) is being created, which serves as an interface to interact with a given contract and provides a list of all methods with its arguments so calls to the contract can be encoded correctly (Solidity ABI, 2019). Once all validating nodes reach consensus on the validity of the transaction, the smart contract code is executed and the contract account is permanently created in the blockchain with its own address, storage file, Ethereum balance and compiled bytecode.

Once deployed to the blockchain, the smart contract code is publicly visible and its functions can be called by all clients of the network. Given the immutability of a blockchain, the smart contract code can not be altered afterwards. However, despite blockchains immutability, smart contracts can be deactivated using a "selfdestruct" function, which can be called on every contract. By deactivating the smart contract, the Ether balance of the contract is sent to a predefined address and the storage and the code is deleted from the state. Messages that are sent to a deactivated smart contract thus won’t call any functions and the containing Ether are lost forever (Solidity Introduction to Smart Contracts, 2019).



```
pragma solidity >0.4.99 <0.6.0;

contract Coin {
    // The keyword "public" makes those variables
    // easily readable from outside.
    address public minter;
    mapping (address => uint) public balances;

    // Events allow light clients to react to
    // changes efficiently.
    event Sent(address from, address to, uint amount);

    // This is the constructor whose code is
    // run only when the contract is created.
    constructor() public {
        minter = msg.sender;
    }

    function mint(address receiver, uint amount) public {
        require(msg.sender == minter);
        require(amount < 1e60);
        balances[receiver] += amount;
    }

    function send(address receiver, uint amount) public {
        require(amount <= balances[msg.sender], "Insufficient balance.");
        balances[msg.sender] -= amount;
        balances[receiver] += amount;
        emit Sent(msg.sender, receiver, amount);
    }
}

```

Figure 3: Smart Contract example, derived from the Solidity Documentation (2019)

As each contract owns a storage file, information can be stored in a key-value format and also altered on the blockchain, whereas all previous states of the information are stored and are publicly visible. Figure 3 illustrates a simple smart contract that issues a token and that allows the smart contract creator to mint an indefinite amount of tokens which can be sent using the "send"-method. *Tokens* are transferable, fungible assets that are created using an Ethereum smart contract. Upon creation of the above smart contract by sending a message with enough gas and the compiled bytecode to the address "0", the constructor method is called, which assigns the sender address to the "minter" address. Afterwards, every client can call the "mint"- and "send"-methods. However, the "require"-operation on line 20 makes sure that the remainder of this method is executed only if the creator account called this method. *Payable* methods require Ether to be send with the message, along with gas (Solidity Introduction to Smart Contracts 2019). When minting and sending the custom tokens, the addresses and the balance values of the owners of the tokens are stored in the *mapping*-type, which is a datatype similar to a hashtable (Solidity Types, 2019). Both *minter* and *balances* values are stored in the persistent storage file of the contract (see figure 2), but can be altered with every valid message sent to the "send"- or "mint"-methods.

2.2.2 Smart Contract Standards

In order to improve the Ethereum platform, an open-source github repository called "*Ethereum Improvement Proposals*" exists, that encourages the collaborative advancement of the Ethereum protocol. This repository features various standards addressing the structure of smart contracts, core protocol specifications and client APIs (EIP 2019), as Buterin sees a need to "standardize certain very common use cases in order to allow users and applications to more easily interact with each other" (Standardized Contract APIs, 2015).

Especially for the popular creation of tokens numerous standards have been proposed, with the "*Ethereum Request for Comment 20*" (ERC-20) standard being one of the most used ones with over 161245 ERC-20 compatible tokens running on the Ethereum network (Token Tracker 2019). In order to create a smart contract that adheres to the ERC-20 standard, the following functionalities have to be implemented (EIP-20 2018):

- A method returning the total token supply. (EIP-20 2018)

- A method returning the balance of another account. (EIP-20 2018)
- A method that transfers a defined currency value to another account. (EIP-20 2018)
- A method designed for withdrawing purposes that transfers a defined currency value from the senders account to a defined, other account. (EIP-20 2018)
- A method that allows a defined account to withdraw from the senders balance up to a defined amount. (EIP-20 2018)
- A method that returns the remaining amount the approved, external account is still allowed to withdraw. (EIP-20 2018)

Even though the contract illustrated in figure 3 allows the creation of a token, it is not compatible to the ERC-20 standard, as it doesn't include the above mentioned required functionalities. With the feasability to issue a digital currency using a smart contract based on the ERC-20 token standard, so called *Initial Coin Offerings* (ICOs) have seen a significant growth in 2017. ICOs are public offerings of newly minted digital currencies in exchange for already existing currencies, such as Ether and are primarily used as a way to crowdfund and finance projects (Fenu et al. 2018, p. 26).

Apart from fungible token, which are each identical in their value, so called "*non-fungible tokens*" (NFTs), which are unique, transferable assets, have been introduced by the ERC-721 standard in 2018 (Entriken et al., 2018). Several decentralized applications, which enable users to collect and trade collectible NFTs with unique attributes, have been developed, such as "Cryptokitties" (2019) and "Decentraland" (2019).

2.2.3 Offchain Data Provider

Morabito (2017, pp. 106,107) differentiates between two types of smart contracts: Smart contracts that are programmed to be executed without external data and smart contracts, which rely on external events and information and which output can vary depending on external data. Hence, he describes smart contracts, which encode rules and conditions that are tied to information that are not included with the invocating message, as being *non-deterministic.*

Since smart contracts can't directly access external data themselves, non-deterministic smart contracts need an entity that gathers outside information (*offchain*) from third parties and submits the gathered information to the smart contract, which then proceeds with its execution (Morabito 2017, pp. 106,107). Several providers exist that offer an oracle service to Ethereum based smart contracts, such as Oraclize (2019) and Chainlink (2019). Besides fetching and delivering data to the smart contract, Oracles can also notify and post information to external systems, such as the outcome of the execution of a smart contract.

2.2.4 Data Storage Options

Storing data on the blockchain can be done by various ways, such as writing to the storage file of a smart contract, by including the data in a message to an externally owned account or by mining a block and including the data as "extra data" in the block header (Wood 2017, p. 5). However, the operation of reading and writing data on the blockchain can be of significant costs, as a document with a filesize of 10 megabytes would cost about 85.76$ to store on the Ethereum blockchain[3]. Alternative options to store data by using an oracle include sending and receiving data from a centralized server, or storing data in a decentralized way, such as using the *Interplanetary File System Protocol* (IPFS) or the *Swarm* service (Mohanty 2018, p. 49). The Interplanetary File System is a system designed to share and store data in a distributed and peer-to-peer way and combines concepts of other peer-to-peer systems, such as Git and BitTorrent (Benet 2014, p. 1). While IFPS was first introduced by Benet in 2014, it is now continually improved upon by the

[3] According to the Ethereum yellowpaper (Wood, 2017), it costs 20000 gas to store a 256 bit string. Therefore, a kilobyte equals 640000 gas. At the time of writing (January 8th, 2019), the average gas price is around 0.0000000134 Ether and 1 Ether costs around 152$. Hence, 10 Megabytes would cost around 85.76$ to store.

open source community (Mohanty 2018, p. 45). Benet went on to create Filecoin, a cryptocurrency project built on IFPS, which enables users to store files by renting unused hard drive space of other network users (Sockin and Xiong 2018, p. 8).

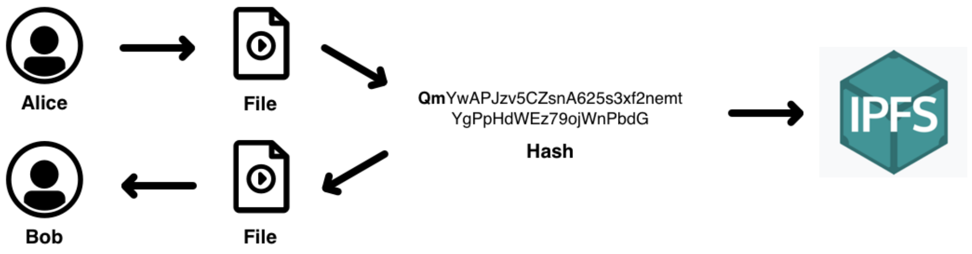


Figure **4**: Storing Data on IFPS (Mohanty 2018, p. 46)

Figure 4 illustrates the functionality of the IFPS system: Alice, who intends to share a certain file, such as an image or a PDF-document, requests IFPS to add the file to the network. By storing the file on the network, which is also referred to as "*pinning*" (IFPS, 2019), a unique hash is being generated from the content of the file, by which the document can be retrieved from the IFPS network. Thus, Bob, who receives the hash from Alice, can now use the hash to download the file that Alice stored in the IFPS network. For privacy reasons, Alice can also choose to limit the access to selected people by encrypting the file with Bobs private key before adding it to the network. This way, everyone who knows the hash of the uploaded file, can download it, but not access the information of the file without decrypting it first using the appertaining private key. In this case, Bob can download the encrypted file and use it by decrypting the file with his private key (Mohanty 2018, p. 46).

Using IFPS, content can be stored in a decentralized way. As the shared file is hosted on multiple nodes, the accessibility of the content is not affected by the outage of a single computer and shared content can not be censored or taken down by without the consent of all sharing nodes. Futhermore, malicious modification of the uploaded content is not possible, as the unique hash ("*content identifier*") of each file gets generated using the content (Raval 2016, pp. 41,42). IFPS is similar to Git in the way the process of updating a file doesn't mean that the file gets replaced (Raval 2016, pp. 41,42). Rather, a new version of the previously uploaded file gets added to the network with a new hash being created, with the previous version of that file still accessible using its hash. Therefore, files cannot be deleted without consent of all sharing nodes and files can only be added using the "ADD"-command and read from using the "CAT"-command (Raval 2016, pp. 41,42).

Similarly to IFPS, Swarm also provides a distributed file storage service with its main purpose to "provide infrastructure services for developers of decentralised web applications (dapps)" (Swarm, 2019). Nodes that agree to rent out their storage get rewarded in Ether. However, unlike IFPS, Swarm is still being developed on testnet (Mohanty 2018, p. 49).

2.2.5 Other DLT Structures

According to Liu et al. (2019, p. 4) ledgers serve to "determine the order of transactions", with each transaction changing the state of the ledger. The specific order partly constitutes the state of the ledger and influences the connectivity of the hashes in the ledgers, as any change to a set of transaction requires computationally intensive hashing. To change any transaction of a ledger, a separate sub-ledger has to be created with a high computational effort, as transactions have to unidirectionally refer to each other using unique hashes. The event of creating a separate ledger is called a *fork*. Forks can happen accidentally, for example when multiple nodes find the fitting nonce in a PoW-consensus at nearly the same time (Zheng et al. 2017, p. 560). In that case, a branch is being created, as two different blocks are added to the previous block. Mining nodes continue with their computation until a longer branch is found, to which all miners of the system then switch (Zheng et al. 2017, p. 560). The abandoned branch is then called a *false fork* (Liu et al 2019, p. 2) with its blocks being called *uncle blocks* in Ethereum. The ability of a ledger to avoid such false forks along with the inability to change the state of the ledger by increasing the cost to alter the transactions are the main objectives of designing a ledger structure according to Liu et al. (2019, p. 4). However, both objectives contradict each other and create a certain trade-off between having fewer conflicts (and also enabling a higher transaction troughput) and a more tamper-proof ledger, as figure 5 by Liu et al. (2019, p. 4) illustrates. Vertices of the blockchain-diagram represent blocks of transactions, whereas each vertex of the partially ordered structures represents a node with a single, unique transaction.

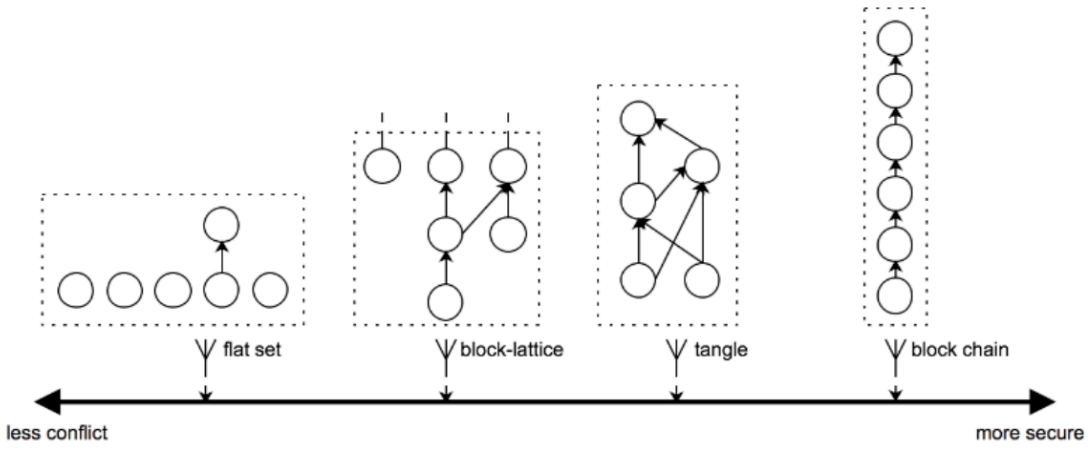
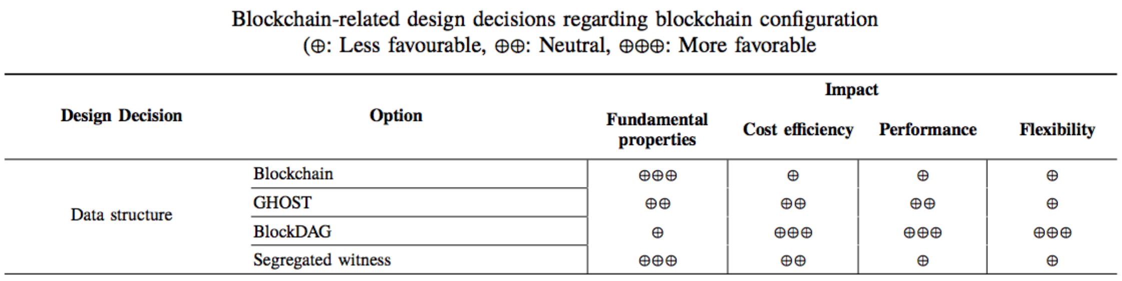


Figure 5: Ledger structure comparison (Liu et al. 2019, p. 4)

A blockchain features a linear order of transactions illustrated by a path graph and thus has a higher tamper resistance than ledger structures with less partial order relations of its transactions. The structures with fewer partial order relations of transactions allow for a better prevention of false forks and also for a higher transaction throughput, but in return feature less resistance against manipulation (Liu et al. 2019, p. 4).

The relationships of the transactions of the *block-lattice*- and *tangle*-structures can be visualized by a directed acyclic graph, as transactions are connected in a partial order without the occurrence of directed cycles (Pervez et al. 2018, pp. 27-29). With the *directed acyclic graph* (DAG) ledger, each new transaction requires the validation of at least one previous transactions before being appended to the ledger (Lee and Choi 2019, p. 4). As the validation and issuance of transactions occurs asynchronously, transactions can be processed at a higher rate than blockchain-based ledgers. Furthermore, as no mining fees exist and the issuing node of a transaction is a validator itself, transaction fees can be significantly lower than in blockchain-ledgers, making the DAG-ledger suitable for the facilitation of micro-transaction in Internet-of-Things (IoT) systems (Pervez et al. 2018, pp. 27-29).

Furthermore, other distributed ledger structures exist, such as the "Greedy Heaviest-Observed Sub-Tree" (GHOST) protocol, which is a modified version of Bitcoin's PoW-protocol (Xu et al. 2017, p. 248) and the "*Segregated Witness*" (SegWit) structure, which is the blockchain-structure that was proposed by the Bitcoin Improvement Proposal 141 (Lombrozo et al. 2018). Table 2 by Xu et al. (2017, p. 249) provides an evaluated comparison between some of the previously mentioned ledger structures based on their "fundamental properties", "cost efficiency", "performance" and "flexibility".



Blockchain-related design decisions regarding blockchain configuration
(⊕: Less favourable, ⊕⊕: Neutral, ⊕⊕⊕: More favorable

Design Decision	Option	Impact			
		Fundamental properties	Cost efficiency	Performance	Flexibility
Data structure	Blockchain	⊕⊕⊕	⊕	⊕	⊕
	GHOST	⊕⊕	⊕⊕	⊕⊕	⊕
	BlockDAG	⊕	⊕⊕⊕	⊕⊕⊕	⊕⊕⊕
	Segregated witness	⊕⊕⊕	⊕⊕	⊕	⊕

Table **2**: "Blockchain-related design decisions regarding blockchain configuration" (Xu et al. 2017, p. 4)

Another DLT structure is the *chain of antichains* (Lee and Choi 2019), which is based on a combination of both the DAG-transaction structure and the blockchain structure in order to profit from the advantages both structures offer. The structure of a typical DAG is maintained, while a second layer almost synchronously manages a blockchain structure, that is based on subset relations among the nodes of the DAG-layer. Various smart contract-enabling platforms exist that are based on a blockchain DLT structure, such as Ethereum, Tron (Tron 2018) and NEO (NEO 2018). Moreover, DAG-based smart contract-enabling platforms emerge, such as "Fantom" (Fantom 2018) and "Vite" (Liu et al. 2019), on which DApps can be built.

2.2.6 Decentralized Applications and their Forms of Architecture

The term DApp is an abbreviation for "decentralized application", which describes an open source software system that makes use of decentralized ledger technology, relies on a consensus mechanism and the utilization of a token (Wessling and Gruhn 2018, p. 45). Similar to non-decentralized applications, a DApp can include both server- and client-side components, but it also employs a decentralized ledger to store certain data. According to Wu (2019), even a single smart contract can be seen as a decentralized application. For an application to be considered decentralized, Johnston et al. (2015, pp. 3,4) propose 3 traits:

- „The application must be completely open source, it must operate autonomously, with no entity controlling the majority of its tokens, and its data and records of operation must be cryptographycally stored in a public, decentralized blockchain." (Johnston et al. 2015, pp. 3,4)
- „The application must generate tokens", which are needed for the interaction with the application. Furthermore, contributions should be rewarded with the respective tokens." (Johnston et al. 2015, pp. 3,4)
- „The protocol of the application can be altered if the majority of its users agree on the change." (Johnston et al. 2015, pp. 3,4)

Raval (2016, pp. 3-8) offers a slightly different characterization of DApps than Johnston et al. (2015) and stresses the consensus creation among nodes and the absence of a single source of failure: Besides being open source, a DApp according to Raval's definition is also characterized by making use of an "internal currency" and a "decentralized consensus" and doesn't have any "central point of failure".

The term "Decentralized apps" can be viewed as a superclass of other sub-types: Raval (2016, pp. 12,13) differentiates between "*decentralized organizations*" (DO), "*automated agents*" (AA), *decentralized autonomous organizations*" (DAO) and "*decentralized autonomous corporations*" (DAC). A "decentralized organization" describes an organizational structure, in which employees participate in the organizations decisions in a democratic way. "Automated agent" is a synonym for an application that runs without any "human intervention" in an autonomous way, which doesn't require any maintenance (Raval 2016, p. 12). A "decentralized autonomous organization", for which Raval names Bitcoin as an example, depicts an organization in which artificial intelligence controls the majority of decisions and human intervention is not necessary (Cai et al. 2018, p. 3). A decentralized autonomous corporations can be viewed as a "subclass of a DAO that pays dividends to its members" (Raval 2016, pp. 12,13).

For further differentiation, Johnston et al. (2015, pp. 4-5) suggest classifying dapps based on their proprietorship of their used blockchain and identify three types:

While a *type I DApp* has its own blockchain, such as Bitcoin, a *type II dapp* makes use of the blockchain of a type I DApp and is considered a protocol that utilizes tokens for its functions. Similar to a type II DApp, a *type III dapp* is a protocol, which is based on a different protocol of another type II DApp and also employs its own currency (Johnston et al. 2015, pp. 4,5). Thus, according to the definition of Johnston et al., a DApp such as decentraland (Decentraland 2019) could be classified as a type III dapp, as it features a protocol, has its own token ("Mana") and makes use of a protocol of a type II DApp for its virtual land auction, which in this case, is Kyber Network (Kyber Integration 2018). Kyber Network (KyberDeveloper 2019) is a protocol that enables decentralized token swaps and uses its own currency ("KNC"), but is built on the smart contract platform of Ethereum, which represents the type I DApp.

Wessling and Gruhn identified identified three reoccuring types of architectural patterns for decentralized applications, which each have a "strong impact" on trust, user experience and security (Wessling and Gruhn 2018, pp. 45,46). However, the identified patterns don't feature all of the characteristics of a DApp the definitions of Raval (2016, pp. 12,13) and Johnston et al. (2015, pp. 3,4) describe. Examples of DApps that are provided with each pattern, for example "Binance" or "Cryptokitties" aren't fully open-sourced applications and only operate partially autonomously, yet interact with one or more blockchains. In their study, Wessling and Gruhn found that each architectural pattern offers a certain trade-off between convenience and trust.

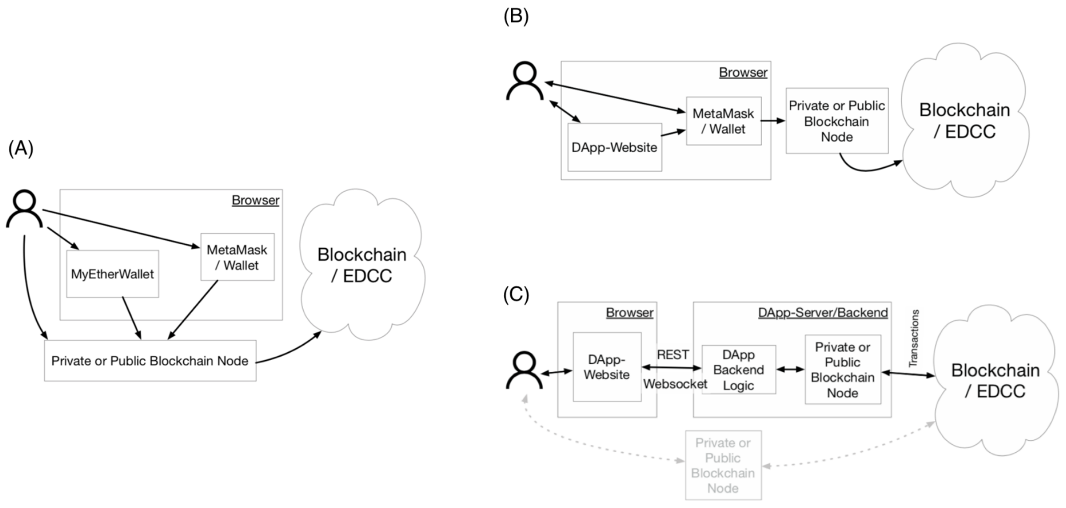


Figure **6**: DApp Patterns A-C (Wessling and Gruhn 2018, pp. 45,46)

The first pattern named "*Self-Generated Transactions*" (A) describes a DApp architecture, in which a user interacts with the smart contract directly by creating and sending transactions on his own. He can choose to either run a node with which he sends a transaction to the smart contract, or he can use a browser interface, such as the web-application "*MyEtherWallet*" (MyEtherWallet 2019). Alternatively, he can also use a browser plugin such as "*MetaMask*" (Metamask 2019) or a dedicated browser such as "*Cipher*" (Cipher 2019), which each include a built-in wallet. However, this interaction requires the user to have a rather high technical understanding, as he has to manually initate the transaction and know the interface description of the smart contract, which can be perceived as inconvenient as it increases the chance of the user making errors. Since this architecture requires a certain degree of technical knowledge and since the private key remains on the user's device, the sense of security and trust is the highest among the three architectural patterns (Wessling and Gruhn 2018, pp. 45,46).

The second pattern called "*Self-Confirmed Transactions*" (B) enables a more convenient interaction for the user, as transactions are generated by the website of a DApp and shown to the user for approval. Thus, less technical knowledge is required than with pattern A. However, a higher sense of trust in the DApp is needed, as the effects of the transactions can appear to be unclear (Wessling and Gruhn 2018, pp. 45,46).

The third architectural pattern named "*Delegated Transactions*" (C) offers the highest degree of convenience for the user, as he doesn't have to manage his private keys and the initiation of transactions himself. The user interacts with the frontend the DApp provides and doesn't need a browser with a cryptocurrency-related wallet or plugin, since the backend of the DApp handles the generation and signing of transactions, which are sent to the blockchain. In comparison, this pattern requires the highest level of trust, because the user doesn't own the private keys with which transactions are being signed. Furthermore, the interaction with the blockchain and the smart contract functionalities can be intransparent for the user (Wessling and Gruhn 2018, pp. 45,46).

In the context of building a hybrid DApp or extending an existing application with blockchain technology, the question arises, which elements of the architecture should be implemented with blockchain technology and which areas benefit the most from being built using blockchain technology. After all, scenarios where the participants can freely interact and trust each other, favor the usage of centralized system due to its cost-effectiveness. Wessling et al. (2018, pp. 43-37) propose a four-step approach to answer this question, by specifically examining the trust and interaction relations of all participants of a given system. In order to illustrate their approach, they describe a scenario of contractors constructing a building, who are supervised by a supervisor and paid by a building owner, who only is in contact with the supervisor (Figure 7).

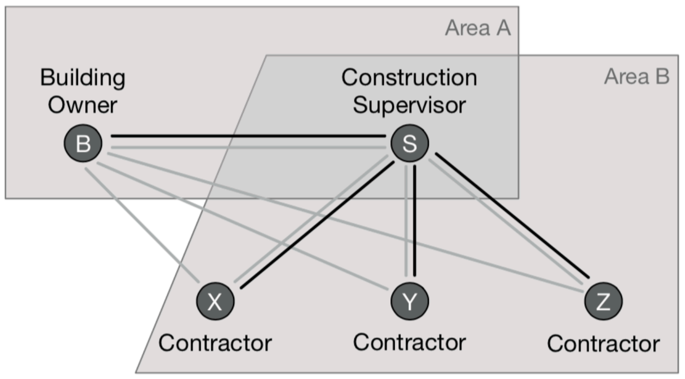


Figure 7: "Emerging areas of the trust and interaction overlay as a first hint towards an architectural draft" (Wessling et al. 2018, p. 46)

Wessling et al. suggest to identify and aggregate all participants that interact with the investigated system first. Afterwards, the trust relations among the participants are examined. According to Wessling et al., trust is a central factor of a blockchain-based application, as blockchain technology "reduces the required trust both into other users and systems" through both transparency and mutual validation of transactions (p. 45). The third step is to identify all interactions between the participants, after which an "architectural draft" is created by using the data gathered from the first three steps. In this final step, certain areas can be detected that describe system elements that can potentially benefit the most from employing blockchain technology. Figure 7 illustrates the approach and highlights the identified areas, along with the trust relations (marked in Black) and interactions (marked in grey). Wessling et al. argue, that blockchain technology is most suitable for areas of a system architecture, in which trust relations exist only transitively and where a need for documentation and payment processing exists. In the given scenario, the participants in area A trust each other, whereas trust between the contractors of area A and the building owner of area B is merely formed indirectly through the supervisor. For this reason, the usage of blockchain technology could connect the building owner to the contractors directly to ensure a transparent documentation of the progress, to allow the participants to create and confirm finished tasks and to enable automatic payments to the contractors (Wessling et al. 2018, p. 46).

2.2.7 The Ecosystem of Decentralized Applications

With blockchain-based applications being viewed as one of the main, recent trends of IT (Marchesi et. al 2018, p. 1), the rising interest in developing DApps has created a competitive landscape in which developers "hurry" to be the first to launch a new DApp on the market (Marchesi et al. 2018, p. 2) Furthermore, it led to an exponential growth of DApps being published, as figure 8 illustrates. Currently[4], more than 2650 DApps exist, of which 2370 are built using the Ethereum platform (State Of The DApps 2019).

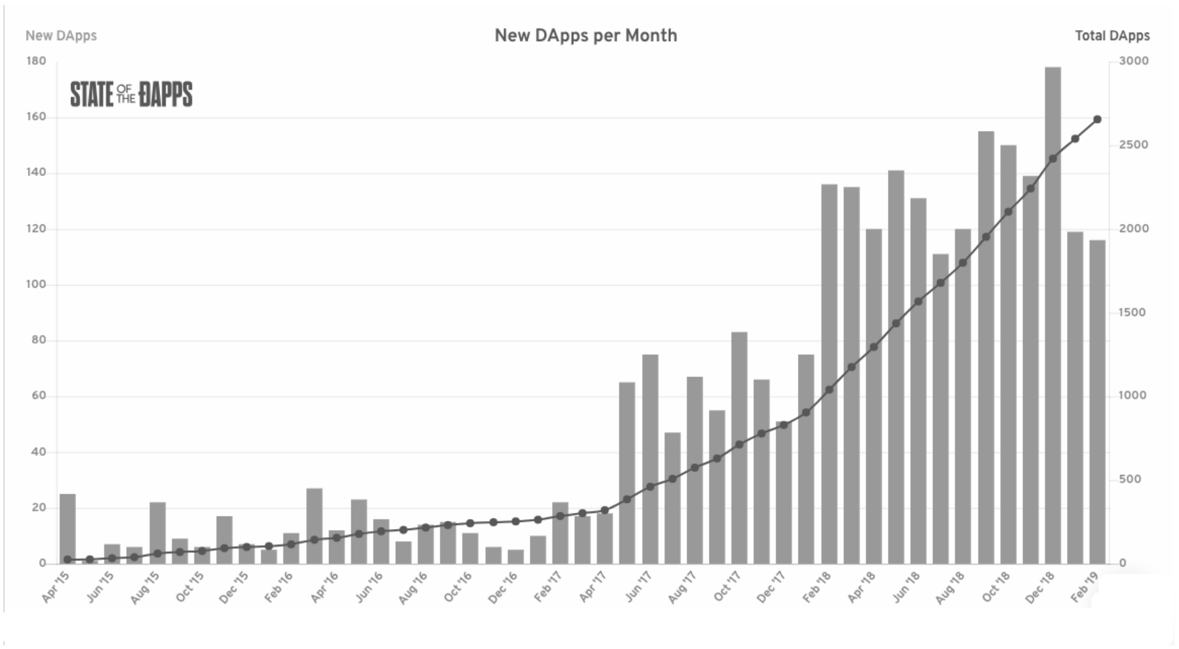
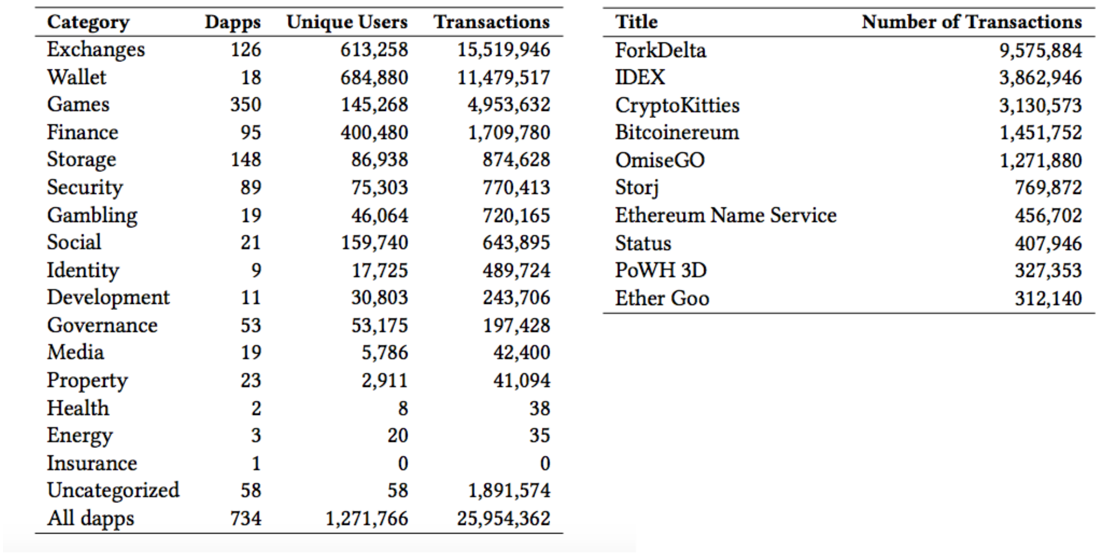


Figure **8**: New Dapps per Month and Total Dapps (State of the DApps 2019)

A study conducted to analyse the distribution and categories among 734 Ethereum-based DApps, revealed, that the categories with the most cumulative users are "wallets", "exchanges" and "finance". Table 3 provides an overview of the distribution of DApps examined in this study. While the categories "exchanges", "games" and "storage" feature the highest amount of DApps, the categories "exchanges" and "wallet" showcase

[4] As of 14th March 2019

the highest amount of transactions (measured in 24 hours).



Category	Dapps	Unique Users	Transactions
Exchanges	126	613,258	15,519,946
Wallet	18	684,880	11,479,517
Games	350	145,268	4,953,632
Finance	95	400,480	1,709,780
Storage	148	86,938	874,628
Security	89	75,303	770,413
Gambling	19	46,064	720,165
Social	21	159,740	643,895
Identity	9	17,725	489,724
Development	11	30,803	243,706
Governance	53	53,175	197,428
Media	19	5,786	42,400
Property	23	2,911	41,094
Health	2	8	38
Energy	3	20	35
Insurance	1	0	0
Uncategorized	58	58	1,891,574
All dapps	734	1,271,766	25,954,362

Title	Number of Transactions
ForkDelta	9,575,884
IDEX	3,862,946
CryptoKitties	3,130,573
Bitcoinereum	1,451,752
OmiseGO	1,271,880
Storj	769,872
Ethereum Name Service	456,702
Status	407,946
PoWH 3D	327,353
Ether Goo	312,140

Table **3, 4**: “Categories and Their Summaries of Transactions” and “Top 10 Dapps in State of the ĐApps” (Wu K. 2019, p. 3,4)

A look at the top ten DApps, according to their amount of transactions (table 4), points out that the exchanges “ForkDelta” and “Idex” are the DApps with the highest transactional volume, which constitute almost all of the amount of transactions of the category “exchanges”. This hints at the finding of the study, that the amount of users and transactions are concentrated on a significantly small subset of all DApps: Around 5% of all measured DApps feature around 94.65% of all users, around 94% of all transactions and around 95.22% of all transactional volume (Wu 2019, p. 5).

3 Inferring Usability Challenges from technical Idiosyncracies

From the perspective of an end-user, both the architectural design of how a blockchain is implemented within an application (Wessling and Gruhn 2018) and the usage of public key cryptography (Eskandari et. al. 2018) can have an influence on the usability. In this section, an overview of the ways, in which specific technical limitations and idiosyncracies of blockchain-based applications can impact the usability and the experience for an end-user, is being developed. Since certain technological traits of a DApp, such as the "robustness" of the blockchain network, can be viewed as positive qualities, challenges, as well as advantages are being described.

Each of the following subchapters describes a group of cohesive usability challenges and qualities that are based on the technological idiosyncracies, which are outlined in the foundational chapter 2. The discussion is guided by an evaluation framework synthesized and presented in the next step.

As no traditional usability- and user experience- inspection methods (Hollingsed and Novick 2007, pp. 1-5) are used in this section, assumptions targeting both concepts are made. Therefore, only a subset of factors that influence the usability and user experience are considered, that are appropriate for this purpose. Prior to investigating the challenges, a framework of appropriate factors gets composed that is based on the taxonomy proposed by Alonso-Ríos et al. (2009, p. 3 ff.) and that can be applied to a certain degree without using a related DApp. The taxonomy by Alonso-Ríos et al. is chosen as a basis, as it combines a variety of existing definitions of usability in an effort to present a hierarchically organized list of attributes that constitute usability.

Factors, such as "user interface aesthetics", that significantly depend on the individual perception and can't be assessed in a generalized manner, are excluded, as well as aspects, that are specifically tied to the respective application and context of use, such as "suitability tor the task" and "context of use" (Alonso-Ríos et al. 2009, p. 3 ff.).

As the framework therefore puts more emphasis on usability than on the user's emotions and perceptions, the first definition of user experience of Bevan (2009, p.3) gets utilized, which considers user experience an extension of the "satisfaction"-component of the concept of usability. Selected aspects of the concept of user experience get incorporated into the framework, as the presented definitions of the aspect of satisfaction in chapter 2.1.1, require a finer granularity for the aspect of "satisfaction" to serve as an evaluation factor in the framework composition. To define the evaluation framework, the taxonomy of usability, as proposed by Alonso-Ríos et al. (2009, p. 3 ff.), including its first and second layer factors, serves as the foundation. According to Bevan's definition, the „satisfaction"-aspect then gets amplified by selected factors that constitute user experience. The frequently examined factors in the context of user experience „enjoyment", „motivation" and „frustration", as identified by Bargas-Avila and Hornbaek (2011, p. 2693), get added. Furthermore, the feeling of „trust" gets appended, because trust is a factor that is commonly investigated in user experience evaluations (Sauro 2015, pp. 68-86) and that is critical for blockchain-based systems (Wessling et al. 2018, pp. 43-37). The concept of *cognitive load*, which has been discussed in the foundational section, correlates with the the aspect 1.3 "Memorability". The evaluation framework is as follows:

Evaluation Framework (EF), based on Alonso-Ríos et al. (2009, p. 3 ff.)

EF 1.	Knowability (Learnability)	
	EF 1.1 Clarity	"The ease with which the system can be perceived by the mind and the senses." (Alonso-Ríos et al. (2009, p. 5)
	EF 1.2 Consistency	Describes the "system uniformity and coherence" (ibd. 2010, p. 5).
	EF 1.3 Memorability	The property of the system that enables the user to remember the elements and the functionality of the system" (ibd. 2010, p. 5).

	EF 1.4 Helpfulness	The means provided by the system to help users when they cannot infer or remember how to use the system" (ibd. 2010, p. 5).
EF 2.	Operability (Effectiveness)	
	EF 2.1 Completeness	"The capacity of the system to provide the functionalities necessary in order to implement the tasks intended by the user" (ibd. 2010, p. 6)
	EF 2.2 Precision	"The capacity of the system to perform tasks correctly" (ibd. 2010, p. 7)
	EF 2.3 Universality	"The extent to which the system can be used by all kinds of users" (ibd. 2010, p. 7).
	EF 2.4 Flexibility	"The capacity of the system to adapt and to be adapted to different user preferences and needs" (ibd. 2010, p. 7).
EF 3.	Efficiency	
	EF 3.1 Efficiency in human effort	"The capacity of the system to produce appropriate results in return for the physical or mental effort that the user invests" (ibd. 2010, p. 8).
	EF 3.2 Efficiency in task execution time	"The time invested by the user in performing actions and the time taken by the system to respond" (ibd. 2010, p. 9).
	EF 3.3 Efficiency in tied up resources	Referring to "material and human" resources. (ibd. 2010, p. 9).
	EF 3.4 Efficiency in economic costs	Describes the "different types of expenses, namely, the cost of the system itself, human resource costs, the cost of the equipment that is required to work with the system, and the cost of consumables" (ibd. 2010, p. 9).
EF 4.	Robustness (Error Tolerance)	

	EF 4.1 Robustness to internal error	
	EF 4.2 Robustness to improper use	
	EF 4.3 Robustness to third party abuse	
	EF 4.4 Robustness to environment problems	
EF 5.	Safety – the “capacaty to avoid risk and damage”	
	EF 5.1 User safety	“The capacity to avoid risk and damage to the user when the system is in use” (ibd. 2010, p. 10).
	5.2 Third party safety	“The capacity of avoiding risk and damage to individuals other than the user when the system is in use” (ibd. 2010, p. 10).
	5.3 Environment safety	“The capacity of the system to avoid risk and damage to the environment when being used” (ibd. 2010, p. 10).
EF 6.	Satisfaction - extended with selected UX factors identified by Bargas-Avila and Hornbaek (2011, p. 2693)	
	EF 6.1 Enjoyment	
	EF 6.2 Motivation	
	EF 6.3 Frustration	
	EF 6.4 Trust	

For clarity, challenges are referenced with an “c”, while advantages are referenced with an “a”. Other aspects, that refer to aspects of the framework, but can’t be clearly assigned to either advantages or challenges, are annotated with an “n”.

3.1 Finality of Transactions

As outlined in chapter 2.2, one of the distinct advantages of utilizing blockchain-technology over non-distributed ledger structures is the high cost of altering historical transactions. As transactions are included in blocks, which are distributed over multiple nodes in the network, the outage of one node wouldn't affect the availability of the transaction records (EF 4.4 a "Robustness to environment problems"). Although transactions, that are appended in a block to a blockchain are deemed immutable (Zheng et al. 2018, p. 357), network attacks, depending on the employed consensus mechanism, exist, that can potentially change the integrity of the ledger (EF 4.3 c "Robustness to third party abuse"). In the case of a "*51% attack*", an attacker creates a fork of the main chain with a PoW-algorithm and then has to have more computational power than the rest of the network combined to create the longest chain, which the other nodes in the network then have to accept, according to the rules of the consensus mechanism (Xu et al. 2017, p. 248). In the process of creating a fork that is longer than the main chain, the attacker is able to rewrite the history of the blockchain and can therefore declare previously valid transactions as invalid (EF 5.2 c "Third party safety").

Khairuddin and Sas (2017, pp. 6506) found out, that Bitcoin users trust in the validity of the transactions is partly constituted by the protocol and "miners' competence and hard labor" (EF 6.4 n "Trust"). Therefore, the risk of a distributed ledger to be attacked, which is influenced by the concentration of mining pools, could potentially lead to distrust and a reluctance to use a blockchain-based application, especially if the persistence of the ledger is perceived as significantly valuable for the user (e.g. in the case of storing valuable digital assets or using a timestamping DApp). Khairuddin and Sas (2017, pp. 6507) also noticed, that the irreversibility of transactions is considered a "risk factor" by Bitcoin users, especially when users deal with potentially dishonest transaction partners (EF 6.4 c "Trust"). As transactions are only reversible through attacks, users can't recover from potential errors they make (EF 4.2 c "Robustness to improper use"). The risk of sending transactions with undesired consequences for the sender is increased, if the sending user interacts with a smart contract of a DApp directly and generates transactions himself (pattern "Self-Generated Transactions"). Depending on the encoded logic of the smart contract, the user could in that case lose all sent funds (EF 5.1 c "User safety") or lose his gas fee in case he made a typing error when specifying the ABI (thus also affecting his level of motivation and frustration – EF 6.2 c "Motivation", EF 6.3 c "Frustration").

The inability to recover from errors also gets posed in the case of storing documents using the IFPS-network (EF 4.2 c "Robustness to improper use"). Even though users can mitigate privacy risks by encrypting their documents, the uploaded files are constantly publicly available, without the user being able to delete the files from the network. However, the censorship-resistance, the inability of others to manipulate data, can be seen as a favorable and unique trait of the IPFS-network and blockchain technology (EF 4.3 a "Robustness to third party abuse").

3.2 Cognitive Load: Usage of Blockchain Terminology

With DApps, the cognitive load required for an user to accomplish task could be significantly impacted by the DLT-technology itself. Concerning the *intrinsic load*, blockchain can be perceived as a topic, that is not easy to understand (EF 1.1 c "Clarity"), especially when having limited related technical knowledge (Dettling 2018, p. 214). The required cognitive load could thus be significant when interacting with DApps, as inherently complex topics can increase the intrinsic cognitive load of associated tasks (Chandler and Sweller 1991, pp. 293-332). Moreover, in the process of interacting with a system, the user develops a mental concept of how the system functions, which helps the user to anticipate future behavior of the system and therefore shapes the way in which the user interacts with the system. According to Nielsen, such "mental models" are constrained by the user's technical background, his previous experience and his "structure of the human information processing system". Furthermore, "learnability", "functionality" and "usability" (Nielsen 1983, pp. 7-14) are aspects that a conceptual model should fulfill for a user to develop a *mental model* that is coherent with the concept of the system that the designer of the system has in mind (ibd., pp. 7-14).

One aspect that can hinder the learnability, which also is a factor that is addressed by the *germane cognitive load*, is the usage of specific blockchain terminology (EF 1.3 c "Memorability"). Eskandari et al. (2018) point out that highly specialized technical language is frequently used in Bitcoin clients, which can be perceived as confusing (EF 1.1 c "Clarity") to novice users, and which highlights the inherent complexity of associated tasks and the difficulty to communicate them to the user in an understandable way. Examples of such terms are "synchronizing with network" (Eskandari et al. 2018, p. 10), "gas price" or "mmenomic phrase" (MyEtherWallet 2019). Another aspect, which affects the learnability and targets the presentation of presented information (*extraneous load*) is the usage of cryptographic concepts and elements, such as hashes. Hashes are rather hard to memorize (EF 1.3 c "Memorability") and rarely used by non-experts, in contrast to passwords (Eskandari et al. 2018, p. 2). However, metaphors (one of Mandel's principles of reducing cognitive load), such as "wallet" or "addresses" are used in the context of blockchain applications and support the learning process of the user (EF 1.1 a "Clarity").

3.3 Usage of Transaction Facilitation Fees and Transaction Throughput

One of the limitations of blockchain technology is the necessity to pay a certain fee for every transaction, which can make applications, which depend on micro payments, such as IoT, inefficient (EF 3.4 c "Efficiency in economic costs") (Pervez et al. 2018, p. 28). The highest average Ethereum transaction fee was measured on the 2nd July 2018, which amounted to around 5.528$ (BitInfoCharts 2019). Such costs can also make DApps, that don't focus on the transfer of ownership of goods, but rely on the blockchain to process or record data, significantly unefficient (EF 3.3 c "Efficiency in tied up resources", EF 3.4).

In the context of healthcare DApps, Zhang et al. (2017, p. 4) point out the challenge of evolving data that is written to blockchain contracts: In the case of a DApp storing healthcare data of patients, data records have to be able to be updated permanently, yet data is not only difficult to modify in large quantities, but also costly to be processed on-chain and every change is permanently recorded.

On the other hand, transaction fees incentivize and motivate miners in PoW blockchains to participate in the consensus making process (EF 6.2 a "Motivation"). Furthermore, transaction fees can make the network more robust (EF 4.3 a "Robustness to third party abuse") by preventing nodes from spamming the network with transactions in an attempt to congest the network (Cai et al. 2018, p. 7). With each transaction on Ethereum, the *gas limit*, which represents the maximum allowed computational steps to be taken and is measured in gas, and the *gasprice*, which is the individually specified cost of each computational step, have to be defined. Therefore, the final transaction fee is calculated by the product of consumed *gas * gasprice.* Generally, transactions with a higher gasprice are preferred by miners and thus get processed faster (EF 3.2 a "Efficiency in task execution time") (Maranhão et al. 2019, p. 5). However, the actual speed a transaction takes to be facilitated (included in a block and appended to the main chain) can only be estimated in advance (EF 1.1 c "Clarity") and depends on several factors, such as the block size, the congestion of the network and the average fee paid in the network. Even though websites exist that estimate the average waiting time in relation to the gas price, the exact waiting time until confirmation is unclear to the user (EF 1.1 c "Clarity"). Furthermore, when specifying a gasprice that is significantly lower than the average gasprice, it can happen that the transaction doesn't get processed at all (EF 4.2 c "Robustness to improper use") (Maranhão et al. 2019, p. 5). With DApps where the user interacts with a smart contract directly, it can also occur, that the gas limit, which the user specified, is lower than the actual executing node needs for the computation, so the call of the smart contract function fails and the paid gas doesn't get reimbursed to the user (EF 4.1 c "Robustness to internal error", EF 5.1 c "User safety", EF 6.3 c "Frustration").

One of the frequently mentioned points of criticism targeting public blockchains is the transaction throughput, as the blockchain design doesn't allow transactions to be processed concurrently. While Ethereum has a higher transaction throughput than Bitcoin with a maximum of 15.62 transaction per seconds (Cai et al. 2018, p. 8), congestions in times of heavy usage of the network can occur, leading to higher confirmation times and higher average gas fees (EF 3.2 c "Efficiency in task execution time", EF 3.3 c "Efficiency in tied up resources", EF 3.4 c "Efficiency in economic costs"). Cai et al. (2018, p. 6) mention the example of Cryptokitties (2019), which at some point accounted for around 30% of all transactions of the Ethereum network, which resulted in around 30,000 pending transactions. Even though the average block time as of 24th March 2019 is about 13 seconds (Etherscan Block Time 2019), users who are used to non-blockchain-based applications, can have expectations of the system providing immediate responses (EF 3.2 c "Efficiency in task execution time") and can therefore potentially become frustrated by such delays (EF 6.3 c "Frustration") (Cai et al. 2018, p. 6).

3.4 Key Management and Absence of Customer Support

In their study of investigating Bitcoin key management approaches, Eksandari et al. (2018) point out various usability issues. While different options exist for the user to store his private keys, existing Bitcoin clients showcase a tradeoff between security and convenience, with no clear superior solution. The available alternatives for the user to store his private keys range from storing private keys on the local device, storing them in password-protected wallets or offline, to storing the keys on a hosted third-party wallet provider (Eskandari et al. 2018, p. 2-4). Like with random system-generated passwords, 256-bit private keys are not values that the user typically memorizes, but instead rather stores somewhere. However, unlike passwords that are used for authentication purposes for web services, private keys can't be recovered (EF 5.1 c "User safety"). In case the user loses his private keys or sends a transaction to an undesired address, there's no central support service that can help (EF 1.4 c "Helpfulness") the user in revoking payments or getting his private keys back. Furthermore, Eskandari et al. found that few resources exist that users can turn to in case they need help (EF 1.4 c "Helpfulness") or have questions (p. 9). While managing his keys, the user not only has to be concerned with not losing his keys, he also has to be aware of possible attacks that aim at getting access to his keys, such as through malware, phishing or theft (EF 5.1 c "User safety"). A solution like using a hosted wallet, which can be provided from an online exchange, might be more convenient for the user, as the user only has to manage his password, which he can recover, but he still might be indirectly susceptible to attacks that target the external wallet provider (Eskandari et al. 2018, p. 4). In that regard, the user has to trust in the integrity and the security of the external wallet providers (EF 6.4 n "Trust"), who usually manage significant holdings of a multitude of users and various cryptocurrency exchanges have shut down in the past, without completely reimbursing its users (Chuhan 2018). Similarly, when using wallet software, the user has to trust in the security measures and the code quality of the software provider (EF 6.4 n "Trust").

Alternatives, such as storing the private key on an offline device, on a paper or signing and exporting transactions from an offline device (*air-gapped storage*), can be more secure in contrast (EF 5.1 a "User safety"), but conversely provide less convenience, as the storage object has to be nearby and transactions are generally less immediate (Eskandari et al. 2018, p. 8).

3.5 Dependence on Tokens and Wallet software posing Barriers to Entry

For an user to be able to use a DApp, the application has to employ certain elements that enable the communication with the blockchain. Depending on the architectural pattern of the DApp (Wessling and Gruhm 2018, p. 46, 47), the front-end application employs a blockchain client and a blockchain client API, with corresponding API libraries as outlined by Pustišek and Kos (2017, pp. 415,416):

- The blockchain client is needed for the communication with the network, as it listens to events and keeps the local chain up-to-date. For Ethereum, *geth* is a commonly used client. (Pustišek and Kos 2017, pp. 415,416)
- The client API exposes the functionalities of the blockchain client, which can be accessed using common web communication protocols, such as HTTP. (Pustišek and Kos 2017, pp. 415,416)
- The client API libraries simplify the development of DApps and the usage of the client API. For Ethereum, web3.js is the commonly used library in that regard. (Pustišek and Kos 2017, pp. 415,416) The ecosystem of decentralized applications, that are built using this library, is commonly referred to as *web3* or *web3.0.*

To be able to send transactions with such DApp, the user needs an interface to access the network. If the user doesn't run his own node (Wessling and Gruhn 2018, pp. 45,46) and if the DApp doesn't follow the "Delegated Transaction"-pattern, he needs to either install a browser plugin such as "*MetaMask*" (Metamask 2019), or a dedicated block-chain-browser or wallet such as "Cipher" (Cipher 2019) first, before being able to access a network (EF 3.1 c "Efficiency in human effort"). Softwares containing the client API libraries might prompt the user to select a network, which can lead to confusion (EF 1.1 c "Clarity"): Apart from the "mainnet", networks such as the *Ropsten-*, *Kovan* and *Rinkeby*-network exist, which are viewed as development and unit testing networks (Mohanty 2018, p. 109).

Furthermore, to initiate transactions, Ethereum DApps require Ether to be spent as gas fees. In case the user doesn't own Ether and the gas fee isn't paid by any third party on behalf of the user, the user has to obtain Ether first, which creates additional effort (EF 3.1 c "Efficiency in human effort"). Moreover, DApps by definition (Johnston et al. 2015, pp. 3,4) make use of its own token, which may have to be spent in order to call certain functionalities of the respective DApp. In that case, the user not only has to own Ether, but also the respective token of the DApp. This can cause an interrupted and frustrating experience for the user, who, upon visiting a DApp's website, learns that he has to install a blockchain client software and buy tokens of the DApp off a third-party exchange website first, before proceeding with any further actions (EF 3.1 c "Efficiency in human effort", EF 6.3 c "Frustration"). Furthermore, the price volatility of Ether and Ethereum-based tokens can pose a usability problem. Due to tokens being publicly traded and the gas fee price being subject to several factors such as the computational steps for the miner to execute a given smart contract, it might be unclear to the user, how the price of the service he pays for gets constituted (EF 1.1 c "Clarity"). As prices are not fixed (EF 1.2 c "Consistency"), the total transactional costs can be hard to predict for the user and can have a detrimental influence on activities such as taxation (Nomura Research Institute 2016, p. 20). Moreover, as the user might be used to pay for web services using a non-cryptocurrency, such as US-dollar, additional cognitive effort might be created when a user wants to calculate the equivalent value in US-dollar he might be more familiar with (EF 3.1 c "Efficiency in human effort").

Khairuddin and Sas (2017, pp. 6499-6510) found out, that the majority of Bitcoin users tend to view cryptocurrency as a store of value and investment and haven't used it to purchase goods. This finding subsequently points out the possibility, that users might be interested in acquiring the token of a DApp as an investment opportunity, but might be reluctant (EF 6.2 c "Motivation") to use the token for the DApp in exchange for the services the DApp provides, as they expect the price to increase in the future. This also suggests a different intention of using the DApp: A user might not be interested in the value the services a DApp provides for himself, but the user might use a DApp with the intention of assessing its potential to drive up the token price.

With software such as Metamask, the user can connect to the Ethereum networks through a gateway, which can be a remote geth client that is hosted on a different computer. In case the user doesn't have his own blockchain client on the same device he uses the DApp with, Metamask by default connects to a remote server that runs a geth client (Maranhão et al. 2019, p. 4), but stores the private keys in the browser. Pattern (b) in figure 9 (Pustišek and Kos 2017 p. 417) illustrates a similar architecture, with the difference that the keys are stored remotely. In case the user intends to run a blockchain client (node) himself, additional effort is required for the user to create and maintain a node (EF 3.1). Pattern (a) in figure 9 (Pustišek and Kos 2017 p. 417) showcases a device that runs the node it connects to internally. Running a blockchain client needs significant computational resources (EF 3.3 c "Efficiency in tied up resources"), as the local blockchain has to be synchronized the blockchain of the chosen network. If the blockchain client is configured to synchronize the full blockchain (*full node*), several gigabytes of transactional records have to be transferred and stored on the local computer (Pustišek and Kos 2017, pp. 416,417), which can take a significant amount of time (EF 3.2 c "Efficiency in task execution time") (Iyer and Dannen 2018, p. 34) and can be "highly unreliable" (EF 4.1 c "Robustness to internal error") (Pustišek and Kos 2017, p. 417). This also poses certain requirements to the device the client runs on, as a hard-drive disk can be too slow, creating a snapshot of the blockchain can take several days (Iyer and Dannen 2018, p. 20) and the internet connection has to to support the constant communication with the network. These requirements can potentially exclude users who don't have the necessary resources to participate in the network (EF 2.3 c "Universality"). However, certain client software, such as *geth,* allow the user to choose from different node options (EF 2.4 a "Flexibility"). While a *full* node not only synchronizes the entire blockchain, but also participates in the consensus process, a *light* node solely maintains current block headers (Iyer and Dannen 2018, p. 20), making it less resource-demanding for the user. Nevertheless, Pustišek and Kos (2017, p. 417) found that running a *light* node can cause events, such as transactions, to be lost (EF 4.1 c "Robustness to internal error").

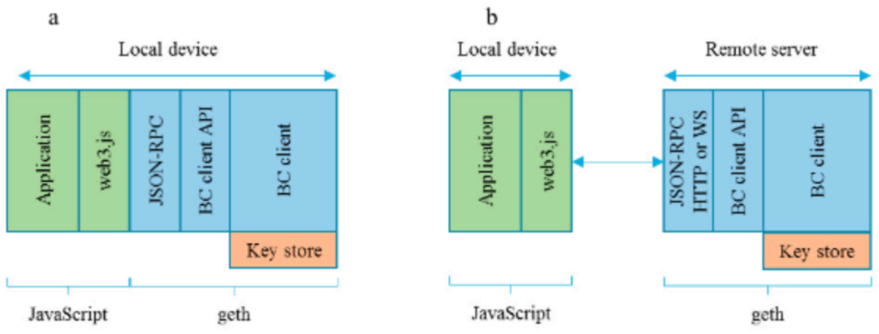


Figure **9**: (a) Stand-alone (IoT) node and (b) remote geth client architectures (Pustišek and Kos 2017, p. 417)

3.6 Transparency of Transactions and Privacy Concerns

The transactions of DApps that are based on public blockchains, such as Ethereum, are transparent and publicly visible. Therefore, blockchain technology can't ensure transactional privacy (Zheng et al. 2018, p. 367), as all actions and balances of all agents, even if new accounts are created, can be viewed and tracked by everyone (EF 1.1 a "Clarity"). At the same time, blockchain technology doesn't reveal the identity of the owner of an account, as public keys can be regarded as pseudonyms of an agent. However, the transparency of all transactions might discourage users to use a DApp (EF 6.2 c "Motivation"), who intent to hide transactional information or who enter sensitive information. After all, financial transactions are considered "highly secret" by many (Kosba et al. 2016, p. 839). With certain use-cases, such as participating in an ICO, *know-your-customer* (KYC) or *Anti Money Laundering* (AML) checks are required by regulatory actors, by which the user gives up his identity and links his identity with his public key (EF 5.1 c "User safety") (Cai 2018, p. 7). The Nomura Research Institute points out a scenario, in which users can be identified based on their transactional history and their medical history (2016, p. 20). Biryukov et al. (2014) present a method of revealing the identity of an agent within the Bitcoin network by linking pseudonymous public keys to the IP addresses that are used to generate transactions, even when users are behind network address translation (NAT) or behind a firewall (EF 5.1 c "User safety"). Using the same address for multiple transactions and sending transactions to the same adresses again can decrease the level of anonymity. Alternative blockchains exist that enable a higher degree of privacy by using techniques such as *mixing* (Zheng et al. 2018, p. 368) or cryptographic protocols such as *zero-knowledge proofs,* however, none of these blockchain projects enable smart contracts to be deployed on their main network (Kosba et al. 2016). Zhang et al. (2017, p. 4) point out a challenge that is posed by the transparency: Even though sensitive information, such as a patient's medical record, can be stored in an encrypted state on the blockchain, current techniques of encrypting data can potentially be ineffective in the future with more advanced decryption techniques (EF 4.3 c "Robustness to third party abuse"). Furthermore, potential implementation errors and vulnerabilities in the encryption process can increase the risk of the user's encrypted data being decrypted (EF 4.1 c "Robustness to internal error").

Khairuddin and Sas (2017, pp. 6503,6504) found that users in general appreciate the transparency of Bitcoin and being able to track payments in real time, which can enhance the level of trust users have in the system (EF 6.4 a "Trust"). Additionally, they learned that privacy and anonymous transactions can create certain challenges: When dealing with anonymous trading partners in over-the-counter (OTC) trades, users were reluctant to send their funds due to transactions being irreversible and having made negative experiences with dishonest traders before, who didn't send anything in return (EF 6.4 c "Trust"). In this regard, Khairuddin and Sas (2017, p. 6504) detected that reputation and de-anonymization can significantly increase trust and credibility. However, exchange websites and DApps are reported to be the most preferred form of exchanging digital goods and cryptocurrencies, as exchanges are governed by financial regulators, which increases the users trust in the service. Furthermore, the historic performance and the future expectation of the performance of an exchange, along with the reputation of an exchange, the amount of trading activity and the ability to contact representatives can contribute to the user's trust (EF 6.4 a "Trust") (ibd. 2017 p. 6505). Even though users value regulation in terms of dealing with an authorized online exchange service, Bitcoin users in general appreciate the relatively low level of regulation of cryptocurrencies, which can create a feeling of control and empowerment (ibd. 2017, p. 6503).

4 Usability Assessment through Cognitive Walkthroughs

The cognitive walkthrough method has been chosen due to its focus on the ease of use and the exploratory learning process of an user with limited previous knowledge of a given system. Chapter 3 points out, that certain aspects of DApps, such as the usage of specialized technical terminologies (chapter 3.2), can be confusing for a user unfamiliar with blockchain technology. Investigating the explorative usage from the perspective of a user with little knowledge of blockchain technology therefore seems to be a logical next step, for which the cognitive walkthrough method can be suitable.

Moreover, the cognitive walkthrough method is "learnable and usable for a computer designer with little psychological or HCI training" (John and Packer 1995, p. 435) and considered efficient in terms of detecting usability flaws, since its execution requires less time and less reviewers than other usability evaluation methods (Gabrielli et. al. 2005, p. 77). The following chapter serves to explain the method of conducting cognitive walkthroughs. Afterwards, the configuration with the DApp selection approach and the results are being outlined.

4.1 The Cognitive Walkthrough Method

A cognitive walkthrough describes the simulation of a user's "problem solving process at each step" (Nielsen 1994 p. 1) of accomplishing a given set of tasks by an expert using an interface to "evaluate the ease with which users can perform a task with little or no formal instruction or informal coaching" (Polson et. al. 1992, p. 742). Furthermore, this method focusses on the "ease of exploratory learning" (Hollingsed and Novick 2007, p. 250), because the tasks should be accomplished by the user exploring the system and guessing the correct actions, rather than relying on previous knowledge of the system (Polson et al. 1992, p. 743). To conduct a cognitive walkthrough, a system (including an user interface and an array of actions), a task scenario and assumptions about the context of usage and target users are needed. To identify potential usability problems, Polson et al. defined three questions that have to be answered at each step during the navigation:

- "Will the correct action be made sufficiently evident to the user?" (Blackmon et al. 2002, p. 1)
- "Will the user connect the correct action's description with what he or she is trying to do?" (ibd., p. 1)
- "Will the user interpret the system's response to the chosen action correctly?" (ibd., p. 1)

During the preparation of the assessment, for each task the initial state of the interface, the action sequence and the user's initial goals have to be formulated (Polson et. al. 1992, p. 748). A goal describes an objective the user intends to achieve and can be written in varying levels of abstraction. A task may be "Log in to the computer", while a goal might specify the user to "start the word processing program" (Polson et al., 1992, p. 749). By defining the required goals for each task, a hierarchical structure of goals gets developed, that can dictate a certain order by using an "and-then" formulation in a documentation. *Actions* describe atomic "physical activities" that are needed to fulfill a given goal.

4.2 The Cognitive Walkthrough Configuration

The goal of conducting user walkthroughs as part of this study is to identify usability issues that are not only distinctive for the specific set of DApps that are examined, but that can also be found with most existing blockchain-based applications. Furthermore, as the the usability of a DApp can be affected by its architecture (Gruhn and Wessling 2018, pp. 45,46), this section aims at examining in which specific ways each architectural pattern does so by inspecting three exemplary DApps. Due to the limited context of this study, three DApps are selected for the investigation, by choosing the category of DApps with the highest amount of daily transactions and by selecting one DApp for each architectural pattern that exists.

In an attempt to both investigate the influence of the architecture of the usability and to capture a subset of the most frequently used DApps of the existing DApp-ecosystem, the following criterias have lead the decision of choosing DApps to investigate:

1. As Ethereum is the most dominant smart-contract platform with the highest amount of DApps being publised, the chosen DApps have to interact with the Ethereum blockchain.
2. Each application has to feature one of the architectural patterns Gruhn and Wessling identified. It therefore has to be a DLT-based decentralized application, yet its code doesn't have to be fully open-sourced and not all of the data and records of operation have to be stored in a public, decentralized blockchain, since the pattern "Delegated Transactions" (Gruhn and Wessling 2018, p. 46) can feature a centralized server that communicates with a distributed ledger on behalf of the user.
3. The overarching category of the chosen *DApps* has to be one of the top three DApp categories according to Wu (2019), based on the amount of daily transactions and the amount of *DApps* being published.
4. To write a coherent set of goals and tasks as part of the cognitive walkthrough and to enable an appropriate comparison of the results of the walkthrough, the DApps have to be relatively similar in their domain.
5. The client of the DApp that communicates with the Ethereum blockchain has to be web-based.

The identification of DApps that fit the mentioned criterias starts with the choice of the category, as its selection discards a significant amount of non-relevant DApps in a top-level way. Table 3 provided by Wu (2019, p. 3) describes the categories "Games" (350 DApps), "Storage" (148 DApps) and "Exchanges" (126 DApps) as most dominant in terms of DApps being published. Furthermore, the categories "Exchanges" (more than 15 million daily transactions), "Wallet" (more than 11 million daily transactions) and "Games" (more than 4 million daily transactions) feature the highest amount of daily transactions. The intersection of these identified categories are "Games" and "Exchanges". As games can have a multitude of rules, narrative styles and ways in which the game is played, the formulation of a coherent set of tasks for the cognitive walkthrough that are applicable to all chosen DApps might become challenging. Therefore (criteria 3), the category "Exchanges" is chosen as the overall category. In the next step, three DApps of the category "Exchanges" are chosen using the registries of State of the DApps (2019) and Coinmarketcap.com (2019) and associated with the three DApp design patterns.

- The first pattern, *Self-generated Transactions*, describes an architecture, in which the user directly interacts with the blockchain through transactions, which he generates himself. Wessling and Gruhn (2018) mention "Idex" as an example, which is the DApp with the second-highest amount of transactions in the study of Wu (2019, p. 4)
- The second pattern, *Self-confirmed Transactions*, delineates a DApp, whereby the transactions are initiated by the DApp website and shown to the user for further verification (Wessling and Gruhm 2018, p. 46). Kyberswap.com (2019) is a DApp that is built using the Kyber protocol (KyberDeveloper 2019) and conforms to this pattern.
- With the third pattern, *Delegated Transactions*, a backend interacts with the blockchain and controls the user's private keys without the user being able to confirm transactions. Wessling and Gruhn mention "Binance" as an example for this design pattern, which is an exchange service that can't be found in the registry of State of the Dapps (2019), but is listed as the exchange with the highest daily transactional volume on coinmarketcap.com (2019).

Having determined three DApps that are subject of this assessment, the core tasks that are used to evaluate the usability are formulated (For a description of tasks see chapter 4.1). Although the investigated DApps all serve the user's goal to exchange digital assets, the underlying, different architectures require the tasks to be formulated in a rather non-specific and high-level way. Therefore, the description of "atomic" *actions* are deliberately excluded. The tasks are as follows:

Task 1. Create a new account to be able to transact.

Task 2. Save the credentials used to gain access to the account.

Task 3. Exchange 5 US-Dollars in Ether for the respective amount of MKR/LRC [5].

Task 4. Confirm that the transaction has been processed successfully.

Task 5. Log out.

In a step between 1 and 3, 10 US-dollar worth of Ether gets send to the new account the user created. As the procedure of this step is the same with each DApp, it is outside of the scope of this evaluation. In related work, Clark et al. (2007) assessed the usability of Tor clients and Eskandari et al. (2018) evaluated the usability of Bitcoin wallets using cognitive walkthroughs. Both studies employ the usability guidelines of Clark et al. (2007, p. 43), which are based on multiple previous usability publications. Due to their applicability in cryptography-related software evaluations, these guidelines are also used in this assessment. The guidelines are as follows:

Guideline 1. "Users should be aware of the steps they have to perform to complete a core task." (Clark et al. 2007, p. 43),

Guideline 2. "Users should be able to determine how to perform these steps." (ibd. 2007, p. 43)

Guideline 3. "Users should know when they have successfully completed a core task." (ibd. 2007, p. 43)

[5] The cryptocurrency *MKR* (MakerDAO 2019) has been chosen as the asset that is exchanged for Ether, as it is an ERC20-compliant token and available on two of the three exchanges. For Binance, the cryptocurrency *LRC* (Loopring) will be exchanged for Ether, as no MKR/Ether-trading pair is available.

Guideline 4.	"Users should be able to recognize, diagnose, and recover from non-critical errors." (ibd. 2007, p. 43)
Guideline 5.	"Users should not make dangerous errors from which they cannot recover." (ibd. 2007, p. 43)
Guideline 6.	"Users should be comfortable with the terminology used in any interface dialogues or documentation." (ibd. 2007, p. 43)
Guideline 7.	"Users should be sufficiently comfortable with the interface to continue using it." (ibd. 2007, p. 43)
Guideline 8.	"Users should be aware of the application's status at all times." (ibd. 2007, p. 43)

In the next step, the cognitive walkthroughs are conducted by the reviewer in the role of the end-user, who has no previous knowledge of the investigated DApps and who hasn't owned any cryptocurrency before. Furthermore, the impersonated user has used consumer web-applications before, but has never used any application that relies on public-key cryptography. The DApps are reviewed using the latest release of the Chrome browser on mac OS. During the review, the formulated guidelines serve as criterias to evaluate the usability of each DApp. The cognitive walkthroughs can be found in the **appendix of this thesis (A-C)**.

4.3 Discussion of Results

Three cognitive walkthroughs have been conducted (enclosed in the appendix: A-C), with each exchange representing one of the three DApp patterns Wessling and Gruhn identified (2018, pp. 45,46). Moreover, Wessling and Gruhn's finding, that each architectural pattern offers a certain degree of trade-off between convenience and trust, can be confirmed when comparing the conducted cognitive walkthroughs: The decentralized exchange "Idex", which is based on "self-initiated transactions", requires the least amount of trust in the DApp, as the user owns the private keys himself, but provides the most issues regarding the usability for novice users. Conversely, the exchange which follows the "delegated transactions" pattern, "Binance", can be perceived as the most convenient exchange for novice users, as the user can access his funds using only his email address and a password, but it also requires the highest amount of trust, as the user doesn't control the private keys to his funds.

Concerning security and key management, all evaluated exchanges make the user aware of the risk of dealing with blockchain assets: Security suggestions, such as picking a strong password (Idex), or displaying a dedicated "safety risk notice" site upon registration (Binance), are provided by all exchanges, along with the information of potential risks involved by not following the suggested security guidelines: "Funds can (and will) be stolen" (Idex). Furthermore, certain security measures are employed by the DApps: Binance offers the support for logging in using an additional "Two-Factor-Authentication" mechanism with a continuously regenerating code and Metamask blurs out the generated seed phrase when creating a new user account to prevent someone else to read the seed phrase.

As the way by which the private keys are managed vary, the options to recover the user's funds also significantly differ with each DApp: Even though Idex offers a "live support"-chat feature, the user is unable to regain control over his funds in case he loses his private key or JSON-file with the related password he created, even if he contacts the support. When creating an account using the browser plugin "Metamask", the plugin displays a sequence of words that serve as the key to unlock and recover the account, which might be easier for a novice user to memorize or write down than a hash. Additionally, Metamask also gives the user the choice to export and view his private key, which gives a novice user, who gradually wants to get more "exposure" to the underlying blockchain technology, more control over his account. One can argue, that Binance provides the most convenient way to regain access to one's funds, as the user doesn't own a private key and can request a new password to be sent to his email account for logging back in to the exchange.

In case the user visits the DApp for the first time and doesn't own any Ethereum account, using a DApp with the Metamask plugin might be the slowest option until the user can initiate his first transaction using the DApp, as the process of downloading and setting up the account takes significantly more time than creating an account with the other evaluated DApps. Setting up an account using Binance might be the fastest way for a novice user, as the user only needs to choose a password and confirm his email address in order to create an account. Creating an account using a password-email combination is probably also the way a novice user might be most familiar with. However, in case the user already owns an Ethereum account, initiating a transaction using Metamask might be the fastest and most convenient way, because the user remains "logged in" within the plugin and doesn't need to enter any password or private key when sending transactions. Furthermore, Metamask lets the user feel in control, as a dialog asking for permission is presented to the user each time a connection to a DApp is created or a transaction is initiated.

A major usability challenge that got revealed during the cognitive walkthroughs is the presentation of financial instruments, purchasing methods (such as "limit order" or "market order") and the chart layout when buying an asset. Even though this issue is tied to the specific use case of exchanging cryptocurrencies, this might confuse a novice user, who might want to purchase a token with the sole intention of using a specific DApp that is based on the token he wants to buy and who doesn't view the purchase of the token as a speculative investment. The need to issue a "buy order" by setting various parameters creates significant effort for the user and doesn't guarantee an instant purchase, in case a seller isn't willing to exchange the token at the specified price or amount. Furthermore, the creation of a "buy order" in an illiquid market can have the consequence of the user purchasing an asset an unfavorable rate. Kyberswap offered the most convenient and intuitive way of making a purchase, as it allows the user to instantly purchase tokens from the smart contract without having sellers setting offers at different asking prices.

Regarding the feeling of familiarity with the interface and the terminology (Guideline 6, 7), Idex appeared to be the most non-intuitive to a novice user, as technical terms such as "Aura Node Opt-in" or "TX verification" are used, which can be confusing. Furthermore, as Idex requires the user to import an account using a private key or a JSON-file, it appears that idex is targeted at users who have a certain, fundamental technical understanding of blockchain technology. In contrast, both Binance and Metamask/ Kyberswap were found to be more intuitive for users unfamiliar with blockchain technology, due to the usage of passwords (Binance) and backup words (Metamask) and the avoidance of technical terms ("secret backup phrase" instead of "mnemonic seedphrase").

A major usability issue was found with the depositing of funds to the smart contract of Idex, which is a necessary step to start trading: The delay between issuing the transaction to the network and the settlement of the transaction caused Idex to not display the sent amount on both the smart contract account and the imported wallet account for a brief moment, which might make the user believe that his funds are lost. A dialog that assures the correct execution of the deposit or that displays information about the status of the transaction, like Kyberswap does (figure 13), can be helpful in reassuring the user and and making him comfortable in using the DApp (guideline 7, 8). Furthermore, the reference to a third-party website (Etherscan 2019) to inform the user about the confirmation status of the transaction can disrupt the user experience.

While conducting the walkthroughs, certain details helped in reducing the mental effort of the user and thereby also reducing the task execution time: Saving the private key in the browser for the next session (Idex and Metamask), as well as setting default values for gas (Idex) and giving the user the option to automatically calculate a certain percentile of the owned Ether amount were found to be helpful in that regard. Furthermore, displaying pictograms that are unique to its associated public keys (Metamask, figure 12) were perceived to be useful in avoiding errors (guideline 4), by helping the user in making sure to send transactions to the intended address. Moreover, Metamasks feature of showing the transactional costs converted to the respective US-dollar amount significantly helped in accomplishing task 3.

Overall, in case the user intends to use a DApp with a non-financial use case, which requires the spending of a specific token, purchasing the respective token first poses a significant barrier to using the DApp, especially if the user has to deal with financial user interfaces, that he might not be familiar with. Furthermore, the price volatility and the effort needed to calculate to the equivalent US-dollar amount can be seen as additional obstacles. Using the respective token of a DApp in an implicit way and not forcing the user to purchase the token first, such as giving the user the option to pay transactional costs with it on an exchange (Binance) indirectly, thus significantly lowers the effort needed to use a DApp. Furthermore, even though the fluctuating value of tokens can pose an incentive to purchase a token and gain interest in a DApp, for users unfamiliar with blockchain terminology, shifting the focus away from the speculative character of tokens and providing a familiar interface with non-technical language that gradually educates the user about the technology and offers the user to control the exposure to the underlying blockchain technology (such as the ability to view the private key with Metamask), can be benefitial to the usability.

This section focused on investigating the usability of exchange DApps. Wu (2019, p. 5) found out, that the DApp categories "exchanges", "wallet" and "finance" feature the most active users. One can argue, that financial transactions have a higher presence in the user experience of these DApps and that DApps in these categories are primarily used to manage financial transactions. Therefore, the question can be raised, whether the identified usability challenges and advantages are representative for other DApp categories, in which the management and emission of financial transactions is not the main objective and in which transactions are issued to transfer data rather than value (Such as DApps in the categories of “social” and “health”). Furthermore, the limited selection of three DApps of one of many categories can make the level of representativeness questionable. Another limitation is the impersonation of the end-user: As the reviewer is familiar with DApps, tendencies to exaggerate the behavior of an inexperienced end-user and to focus on issues that were already known to the reviewer might have affected the cognitive walkthrough.

Considering the high usage of DApps in financial categories, certain questions arise:

Is the adoption of DApps in general primarily driven by the user’s endeavor to achieve financial gains? Do users initially find out about DApps through their effort to find a speculative investment opportunity or because the DApp solves a problem they face? In what way does the speculative nature of including a token influence the usability – especially in regard to DApps with a non-financial focus? These questions will be addressed in the following semi-structured interviews of the next chapter.

All identified challenges and favorable traits (advantages) have been added to the evaluation framework, which **can be found in the appendix (D)**.

5 Conducting semi-structured Interviews

Drawing on the the experience of DApp developers, the aim of this chapter is to empirically investigate the impact of the previously identified usability challenges, as well as to work out additional usability challenges and related recommendations on how to solve the mentioned issues. Furthermore, the interview partners are asked about traits of DApps, that can contribute to a favorable usability.

The semi-structured interview method has been chosen due to its flexibility and versatility (Kallio et al. 2016, p. 2955), which can be benefitial to yield detailed information beyond the predetermined interview questions. The structure of the interview can thus be adjusted dependent on the participant's responses and the interviewer can ask follow-up questions to gain a better understanding of the specific talking point. This is significant for this research approach (as outlined in chapter 1), as the previous findings are extended with empirical data from the perspective of a developer in this section. Furthermore, follow-up questions help to identify details that might otherwise be unregarded, but that can have a potentially significant impact on the usability. Lastly, the semi-structured interview method is considered suitable for investigating "complex issues" (Kallio et al. 2016, p. 2959).

According to the framework developed Kallio et al. (2016, pp. 2962, 2963), formulating an interview guideline can help directing the conversation to research topics and can promote confirmability and trustworthiness of the overall research. Additionally, an interview guide can evoke answers that are spontaneous, in-depth and unique (ibd. 2016, p. 2960). The next subchapter describes the interview configuration and the interview guideline. Afterwards, general results of the interviews are discussed, whereby the results of each interview gets briefly summarized, while having the overall research questions, as outlined in chapter 1, in mind. In chapter 5.2.2, a list of mentioned challenges gets presented and in the subsequent chapter 5.2.3, the recommendations of the interviewees to achieve a more approporate usability, get outlined.

5.1 Planning and Conducting the Interviews

Kallio et al. (2016, p. 2960) suggest including questions that are clearly worded, participant-orientated and not leading. Furthermore, two types of questions are generated: Questions that address "main themes" (marked as **MQ**) and follow-up questions (marked as **FQ**) (Kallio et al. 2016, p. 2960). The interview guideline is as follows:

1. Getting to know the interview partner
   - **MQ1.** May I record the audio of interview?
   - **MQ2.** Could you provide a brief overview of your work and the product you're working on?
     - **FQ1.** What is your experience in the field of developing and deploying DApps?
     - **FQ2.** How do you gain feedback from end-users regarding usability?
2. Challenges and Recommendations
   - **MQ1.** What is your definition of usability?
     - **FQ1.** Do you agree with the criterias of usability as presented in the framework?
   - **MQ2.** Do you consider usability a major issue to achieving a higher adoption/ acceptance of DApps?
   - **MQ3.** What do you consider to be the main usability challenges of DApps?
     - **FQ1.** What mentioned challenges do you agree with the most?
     - **FQ2.** Could you describe a specific example? (i.e. scenario or DApp)
     - **FQ3.** How did you realize that this is an issue users face?
     - **FQ4.** If you're directly working on usability problems: How do you measure and keep track of the usability?

FQ5. How do the users cope with this challenge?

MQ4. Do you disagree with some of the usability challenges from the list?

MQ5. Concerning the challenges you mentioned – what do you recommend to overcome these challenges?

FQ1. Could you describe how you tackle this usability issue in your current work?

FQ2. Can this challenge be solved on a technical level?

3. General questions regarding selected previously identified challenges

MQ1. According to current statistics, DApps of the categories "exchanges", "wallet", "games" and "finance" have the most active users. Considering the high usage of DApps in financial categories, is the adoption of DApps in general primarily driven by the user's endeavor to achieve financial gains? Do users initially find out about DApps through their effort to find a speculative investment opportunity or because the DApp solves a problem they face?

FQ1. As financial DApps experience a high usage – should DApps in general make the user aware of the underlying blockchain technology and foreground it – or should the technology rather be unnoticed by the user?

FQ2. If the technology should have a high presence: In what way should the user learn about the technology when using the DApp?

MQ2. In what way does the speculative nature of including a token influence the usability – especially in regard to DApps with a non-financial focus?

FQ1. Does the price volatility create a reluctance to use DApps?

The interview guide starts off by asking the interviewee about his perception of the criticality of the concept of usability in relation to achieving user adoption. This question can be helpful when assessing and comparing the responses from all interview partners. The interview participants were recruited via Twitter (2019) and Discord (2019). All interviews have been held and recorded using video conferencing software (Zoom 2019) and transcribed afterwards. Due to the dynamics of the conversations, the interview guideline hasn't been followed in a linear and strict manner, leaving some questions unasked. Therefore, follow-up questions have been composed based on previous answers to get more detailed information of a mentioned topic. In total, 3 interview partners, who are all founders of companies in the DApp-ecosystem and DApp developers, have been interviewed, with the interviews ranging between 34-45 minutes. The transcribed interviews **can be found in the appendix (F)**.

5.2 Interviews Results

Interviewee A states that certain limitations and approaches, such as the emphasis on decentralization in the current, "fragmented" ecosystem, can hinder usability and adoption: He criticizes the industry's focus on security, privacy and decentralization, which stands in the way of abstraction (A009-012), as he considers the process of abstraction to be centralized in many ways, but necessary to provide value to users. Abstracting elements away, especially financial mechanisms, is important in his eyes, especially for mainstream audiences (A012-016), because in his opinion, users shouldn't need to know how blockchains work. Rather, the user experience should follow patterns that the user already knows from other contexts, for example when describing what gas fees are. He claims to be following a principle he calls "progressive disclosure", by which a user gets gradually exposed and educated about the underlying mechanisms as the "stake" of the user increases (A152-162).

He also thinks that the industry's approach until now, to build the technological layer first, before having the target user and the product in mind, has to shift towards a more "product-centric approach", which includes building out detailed "personas"

(A016-029) of users, because different types of users exist, that have different needs and requirements, especially in terms of security: Institutional users might put a higher emphasis on security than non-institutional users, which can mean that the application has to be less decentralized to provide a higher security (A016-052). Furthermore, he is convinced that making the onboarding flow more seamless by not having the user download a DApp browser or wallet before being able to use a DApp can contribute to a higher adoption of DApps and suggests the Fortmatic SDK (2019) as a solution (A036-39). He considers the attribute of Decentralization more important than the transactional speed for a DApp (unless it's a decentralized exchange – A050-052) and thinks that layer-2 solutions, such as state channels, can help with speeding up transactions. He sees a challenge in acquiring cryptocurrencies before using a DApp, which gets reinforced by regulatory aspects (A065-079). Additionally, he considers conveying value to the user outside of speculation purposes as a challenge (A074). Furthermore, he thinks that the volatility of currencies can hinder adoption, as deflationary currencies like Bitcoin provide no incentive to spend the currency (A178-183). He views stablecoins, such as DAI, which has an interest rate, as an alternative, but he thinks that it takes time for mainstream audiences to build trust into stablecoins and in decentralized finance (A088-090). He also believes that there are a lot of "fragmentations" and "inconsistencies" regarding the user experience in the industry, and refers to the early days of the internet. Users have to learn various ways to interact with the blockchain, which hinders adoption. He finds that the user experience improves together with the infrastructure and that collaboration of developers is important, before clear and necessary standards, especially concerning the terminology, the interaction and the technology emerge (A103-113). However, he holds the view that the user experience of a product can suffer and become complicated if the product is open sourced, which he considers to be controversial (A120-126).

Interviewee B considers usability as a problem, as it hinders the fast process of experimentation (B001-003), even though he thinks that the usability infrastructure is built out in a sufficient way (B005-006). He finds that scalability and the process of onboarding are "solved" (B018) and subsequentally mentions solutions, such as gateways to buy cryprocurrencies with non-cryptocurrencies ("fiat on ramps"), the deployment of smart contracts using the Counterfactual framework when closing state channels, meta transactions, the existence of usability best practices and the option to authenticate with a service called Portis (B001-009). He rather sees the challenge in finding and creating valuable products, use cases and figuring out the unique valuable traits of web3.0 (B218-220), which might be an approach to overcome usability issues: "The more valuable a product is, the more users are willing to go through friction" (B011-012). He considers decentralized finance (DeFi) successful in terms of raising funds ("ICOs") (B043-047, B219), but thinks that web3.0 has more fields of application to offer than the financial space, such as in the field of gaming, enabling social experiences and collectible NFTs (B063-065). However, he claims that it's hard to make out usability challenges if one is already familiar with using DApps (B111-112). He believes that blockchain technology will be as omnipresent as the internet today and that most users will know how to use the technology, but won't have a deep understanding of it, similar to the internet (B095-106). Like interview partner A, interviewee B emphasizes the existence of different types of users and the need to provide a more targeted user experience, as a user with a high account balance might have a higher need for security in relation to a user with less funds (B135-139). When being asked about ways to improve usability, he mentions the messaging DApp "Peepeth", which uses stateless contracts, state channels, IPFS and batched transactions (B144-149). Regarding scalability solutions, he prefers state channels over sidechains for security reasons, even though he considers the implementation of state channels to be "very difficult" and time-intensive and thinks that sidechains are more flexible in relation (B153-163). Concerning the adoption of DApps, he thinks that usability is frequently mentioned as an issue by the developer community and underlines the importance of finding ways to deliver valuable DApps and create value through web3.0 technology (B210-220). The only way to get there, in his opinion, is through experimentation and building "things that we might think are interesting" (B233-234).

Interviewee C thinks that "a lot" of usability challenges exist (C046), especially in the field of blockchain-based games. He states that software like "Metamask" requires the user to know about blockchain technology and security practices, such as remembering seed phrases and not publishing the private key by mistake (C020-022). Moreover, he thinks that most of the current blockchain games aren't building on layer-2 technologies, and will therefore not provide the same gameplay speed and experience gamers are used to from non-blockchain technologies (C046-048). In his opinion, the user should not know about the underlying blockchain technology when playing a blockchain-based game (C048-049), unless purchases of tokens can be made, which might be more expensive than in non-blockchain games (C060-062). In this case, he believes that users should be educated gradually about the technology. In addition, he thinks that implementing services like Wyre to purchase tokens using a credit card might be a process that users, that are new to DApps, are familiar with from non-blockchain games (C067-070). Like interview partner B, he stresses the utility of issuing tokens as a way to raise funds as a developer (C078-081).

Like interview partner B, he views services like Fortmatic (2019) and Portis (2019) to be significantly helpful for authentication and transaction signing when using DApps: He explains that Portis allows private keys to be generated from the user's email address and password, which are stored on the client-side, so the user can log in from any device and browser without having to know his private key (C022-029). Like both previous interview partners, he doesn't view scalability as an issue due to layer-2 technologies and explains a game scenario of using a Matic sidechain with the Ethereum main chain to achieve a gameplay with transactions taking less than one second (C031-036). This way, he also points out the possibility to design interactions between multiple blockchains in a creative way (C049-055).

Like interviewee A, he thinks that price volatility of tokens can influence the usability in terms of the understanding of the user and potentially creating reluctance to spend the tokens (C087-089). He suggests the usage of stablecoins, such as DAI as a possible solution. Regarding the usage of gas, he advocates hiding terminologies like "gas" and paying the gas fees on behalf of the user as the developer ("gas relay" – C113-117).

5.2.1 Mentioned Technologies as Solutions to Challenges

During the semi-structured interviews, several technologies have been named, which have not been explained in the theoretical foundation before (chapter 2). In this subchapter, these technologies get addressed to support the understanding of the following chapters.

Approaches to enable a higher scalability are commonly divided into different categories: Besides the network infrastructure layer (*layer-0*) and the layer of the consensus protocol (*layer-1*), approaches that aim at facilitating a higher transaction throughput beside the main blockchain (*off-chain*) are referred to as *layer-2* technologies (Jourenko et al. 2019, pp. 1-2). For Ethereum, layer-2 technologies include *state channels* (Jourenko et al. 2019, p. 2), *Plasma* (Poon J, Buterin V 2019, p. 1) and *Truebit* (Teutsch J, Reitwießner C 2019, pp. 2-5).

Plasma is a framework by which a hierarchical tree-like structure of *child chains* (Johnson S et al. 2019, pp. 1-2) can be created, which are connected to Ethereum's blockchain through smart contracts, regularly report back to Ethereum's main blockchain and on which transactions can potentially be facilitated faster than on Ethereum's main blockchain. Various adapted implementations of the Plasma framework exist, such as Loom's Plasmachain (Johnson S et al. 2019, p. 2) and Matic Network (Kanani J et al. 2019), which can be seen as *sidechains*, which are bridged to Ethereum's main blockchain.

A state channel allows a set of users to transact *off-chain* with each other without having to pay gas fees and with instant finality (Coleman J et al. 2018, p. 1). Participants of a state channel only need to send transactions to the blockchain twice, when creating and closing the state channel (Chernyaeva A 2019, pp. 2-3). When a state channel gets created, users lock their funds in a *multisignature* wallet (referred to as *"state deposit"*), together with a ruleset that is used for interpretation of the state and agree to each other to regularly update the state of the channel. Upon creation of the channel, users can then transact with each other by signing and sending each other messages directly *off-chain*. When closing the state channel, the most recent state update is sent to the multisignature wallet as an on-chain transaction and the funds get distributed according to the ruleset and the most recent state. Since the wallet requires the signatures from all participants of the state channel (multisignature), only the most recent state that all participants agreed on and that doesn't get countered by a more recent state, gets enforced on-chain (Coleman J et al. 2018, p. 1).

Besides payments, smart contracts can also be executed off-chain in a state channel. Furthermore, different approaches and frameworks exist to create *generalized state channels*, in which users can install multiple smart contracts of DApps off-chain without having to send any transaction to the blockchain to deploy or execute the smart contracts (Gudgeon L 2019, p. 5). Participants of such a generalized state channel can deploy (*"Counterfactual instantiation"* - Coleman J et al. 2018, p. 17) and concurrently (Gudgeon L 2019, p. 5) execute multiple smart contracts within the state channel off-chain (Dziembowski et al. 2019, pp. 2-3). This way, the participants compute the most recent state using the smart contract themselves and check on the correct computation by signing and exchanging the most recent state (Dziembowski et al. 2019, pp. 3). To finalize the outcome of the transactions when closing the state channel, the *counterfactually deployed* smart contracts within the state channel have to be also then deployed on-chain, which decide over the correct distribution of the state deposit (Coleman J et al. 2018, p. 17). As the on-chain address of a smart contract is not known at the time of deployment in the state channel (*counterfactual instantiation/deployment*), but needed for its execution, a global registry smart contract, that is deployed on-chain, can be accessed, that delivers the future on-chain smart contract address (Coleman J et al. 2018, p. 17).

5.2.2 Identified Challenges

A list of all named challenges gets outlined in this section. Aside from challenges that affect the usability for the end-user, some issues were mentioned by interviewees that stem from the general state of the usability infrastructure and influence the development progress of the industry, such as that usability can hinder the process of rapid experimentation (B001). Even though these issues can indirectly have an influence on the end-user usability, these types of challenges are not addressed in the following list, as they don't directly correspond with the aspects of the evaluation framework. Each challenge gets marked with "CH". The list is as follows:

CH1	Focus on security, privacy and decentralization can hinder abstraction – abstraction is necessary when hiding the blockchain technology from the user (A011-016)
CH2	The industry's approach to build technologies first without having a clear view of the target user (A019-021)
CH3	Acquiring cryptocurrencies with non-cryptocurrencies is considered difficult (A070)
CH4	Convincing value to the user outside of pure speculation purposes is considered challenging (A073, B019)
CH5	Price volatility of cryptocurrencies can hinder adoption (A085)
CH6	Reluctance to spend cryptocurrencies on DApps due to potential price increases (A178-179)
CH7	Distrust in stablecurrencies (A085-86)
CH8	A multitude of different ways of interacting with the blockchain exists and no clear standard exists yet – the user has to learn different user experiences, which hinders adoption (A106)
CH9	Open source development of products can lead to complicated user experiences (A120-126)

CH10 Effort to download & install software like Metamask can result in significant bounce rates (B003-005)

CH11 Using the Ethereum's blockchain for single transactions can be significantly cost-inefficient (B145)

CH12 Software like "Metamask" requires knowledge about blockchain technology due to blockchain terminologies and security practices (C020-022)

CH13 Most current blockchain-based games don't utilize layer-2 technologies, thus the gameplay experience is unlike non-blockchain games (C046-047)

CH14 Designing the purchasing process when buying tokens with non-cryptocurrencies (C067-069)

In the next step, these identified challenges get added to the existing table of challenges, which **can be found in the appendix (D)**.

5.2.3 Identified solution-orientated Approaches

A list of all mentioned recommendations, that contribute to a favorable usability, gets presented in this chapter. Each recommendation gets marked with an "R". Explicitly named recommendations, that correspond to previously named challenges, are referenced with "CH" (e.g. "CH2"). Recommendations that don't directly affect the usability for the end-user, such as the recommendation to go through a phase of "experimentation" to find valuable products and use cases to build (B234) or the recommendation to develop a common understanding of the unique and valuable traits and use cases of web3.0 to be able to build valuable products (B028-031), have not been included. The explicitly named recommendations by the interviewees are as follows:

R1 Building DApps and web3 products while having a clear understanding of the targeted end-user (A033, CH2)

R2 Using layer-2 technology to accelerate the transaction speed, instead of facilitating all transactions on the main chain (A044-045, C046, CH11, CH13)

R2.1. Using state channels (B154-155)

R2.2. Using sidechains, that can be built using the Plasma framework, such as Matic network (C031-036, B159-163, B174)

R3 Allowing a paradigm shift towards abstraction to happen (for a better understanding for the user), even at the potential expense of decentralization, pricacy and security (A059-060, A197-198, C113, CH1)

R4 Conveying value to the user that's outside of speculation (A072-073)

R5 Foster trust in stablecurrencies (A086)

R6 Promote collaboration among the industry to help the evolvement of industry-wide standards of interacting with the blockchain (A112, CH8)

R7 Development of industry-wide standards of user experience, technology, terminology and interaction with the blockchain (A130-132, A147)

R8 Gradually exposing the user to the underlying blockchain technology (A152-162)

R8.1. …as the "stake" of the user increases (A152-162)

R9 Utilizing stablecoins, such as DAI, to mitigate the issue of volatility and the possible reluctance to spend tokens on DApps (A173-183, C087-089, C119-121)

R10 Hardware key encryption can improve the user's security of his funds not being stolen (A214-226)

R11 Using authentication and key management solutions that are familiar to a novice user and reducing effort by not forcing the user to download additional software like Metamask, for example using a email & password combination (Portis 2019) or the phone number (Fortmatic 2019) (B118-119, C019-029, A042)

R12 Paying the gas fees on behalf of the users (*Meta transactions*) (B121, C114)

R13 Deploying a smart contract using *generalized state channels* increased flexibility and cost-efficiency (B122)

R14 Different onboarding experiences in regards to target users (B138-139)

R15 Utilize stateless smart contracts for decreased gas costs, so data can get accessed from the messages, which maintain the state, rather than the contract (B145-146)

R16 Use IPFS to store large amounts of data outside of the smart contract (B147)

R17	Batching transactions to reduce gas fees (B147-148)
R18	Designing creative ways and user experiences when interacting with sidechains (C049-055)
R19	Implementing non-cryptocurrency to cryptocurrency payment services due to familiarity and convenience (C067-068)
R20	Develop protocols and tools in an "open source"-way, but not products and user experiences, to prevent "complicated" experiences (A124). (CH9)

6 Inferring Recommendations and Requirements

In this chapter, recommendations are synthesized based on the identified challenges of chapters 3, 4 and 5 and appended to the list of explicitly mentioned recommendations of the semi-structured interviews, which can be found in chapter 5.3.3. To synthesize recommendations and to elaborate on previously identified challenges, the overarching categories of chapter 3 get addressed, as the identified challenges from the chapters 4 and 5 can be included in these categories. Recommendations, that have already been explicitly mentioned in the semi-structured interviews get briefly addressed in this chapter together with synthesized recommendations. At the end of this chapter, a list of all synthesized recommendations, that have not been already mentioned in chapter 5.3.3, gets presented. A table with both synthesized recommendations and recommendations of the interviewpartners **can be found in the appendix (E)**.

The finality of transaction (chapter 3.1) poses a significant challenge to the usability, as users can't recover from errors and revoke transactions, which also applies to data uploaded to the IPFS network. Since attacking the network to reverse transactions can't be an option due to cost efficiency and putting the integrity and trust in the blockchain at risk, one solution to this challenge could be to prevent unintended transactions from happening in the first place. Errors might happen in situations when the user has to specify transactional information, such as the gas price or the recipient's address. DApps that follow the patterns "Delegated transactions" or "Self-confirmed transactions" (Wessling and Gruhn 2018, pp. 45,46) relieve the user from certain tasks of transactional specification and can thus prevent errors. However, this pattern gets accompanied by the trade-off of the user having to trust the DApp provider (Wessling and Gruhn 2018, pp. 45,46). This corresponds with the recommendation of interviewee A (A013) to abstract elements, which otherwise might require mental effort for the user and which can be sources for errors, away. Displaying dialogs that show a summary of the transaction and ask the user for confirmation (Figure 12), as Metamask (2019) does, can also help in reducing errors. Additionally, Metamask's feature of providing a visual and unique representation of the address, can support the user in making sure he entered the intended address. Another way of reducing errors is making use of the *Ethereum Name Service (ENS)* (ENS 2019), which can assign human-readable names to hash-addresses, so the user might only have to enter “alice.eth” as the recipient of a transaction instead of “0x787192fc5378cc32aa956ddfdedbf26b24e8d78e40109add0e-ea2c1a012c3dec” (ENS 2019).

The topic of *cognitive load* (chapter 3.2) deals with the mental effort for the user to learn to interact with DApps and to learn blockchain terminology that is being used within DApps. Even though the concept of the *mental model* points out the necessity of the user understanding the respective system he interacts with, Maxwell et al. (2015, p.2) suggest it is not necessary for a user to understand the underlying technology to be able to use a blockchain-based system. An approach to support the learning process of the user could therefore be to avoid using blockchain specific language (“fixing terminology”, A147-148) and to rather use metaphors that a user, who is new to DApps, might be familiar with, such as “wallet” (Eskandari et al. 2018, pp. 9-10). The abstraction of technical details can thus be benefitial for reducing the cognitive effort for the user (“you don’t need the user to know that much information to start” A160-162, “the best way […] is to not show them”, C113). The recommendation of interviewee A, to then “progressively disclose” (A152) information about the underlying blockchain technology to the user, might also help the learning process of the user and potentially increase the adoption of a DApp. Therefore, one could imagine DApps that could automatically adapt their user interfaces to the knowledge, the desired security preferences and the desired exposure to blockchain technology the user has. Metamask’s feature of automatically setting the gas price according to the user’s speed preferences and the current average gas price (Metamask: Sending Transactions, 2019), is an example of abstraction and lowering the user’s cognitive effort. Another feature of Metamask that supports the understanding of the user is converting and displaying the amount of cryptocurrency in a familiar non-cryptocurrency, such as US-Dollar (Figure 12). Using domains registered by the Ethereum Name Service (ENS 2019) can not only reduce potential errors, but also decrease the cognitive effort and increase the memorability of addresses. Furthermore, standards in the way users interact with DApps could reduce the cognitive effort, because a user doesn't have to relearn the interaction each time he visits a new DApp (A103-113). In this regard, libraries of UI-elements and DApp components, such as "Dapparat.us" (Griffith A., 2019), that developers can implement and reuse in their DApps, can also support this process. Another way to lower the effort of the user is to enable subscriptions in DApps (EIP-1337), which allows for recurring payments that the user has to manually confirm only once.

Regarding transactional fees and the transaction throughput, various recommendations have been stated by the interviewees. Requiring the user to purchase Ether before using a DApp is considered a challenge by one interviewee (A070-071). One recommendation in this regard has been to have the DApp developer pay the gas fees for non-*payable* method calls of smart contracts on behalf of the user (C113-121) through *gas station networks* (EIP-1613, 2019), so the DApp user doesn't need to acquire Ether first, which thus removes the barrier in terms of effort to using a DApp. The onboarding SDK Portis (2019), which enables the user to log-in using an email/password combination, allows the DApp developer to implement this functionality into a DApp with a couple of lines of code. Portis and Fortmatic (2019) are both wallet software, which doesn't have to be downloaded and installed on the device of the user and might therefore decrease the user's entry barrier to using a DApp for the first time. Moreover, they offer ways of authentication, which users that are new to DApps might be more familiar with. To further the adoption of DApps, giving out tokens to DApp users for free as rewards or loyalty points to incentivize and motivate users, which are commonly referred to as *airdrops* (Victor F, Lüders B K 2019, p. 3), could potentially help DApp developers with the acquisition and retention of DApp users. This could also mitigate the issue of users being reluctant to spending on a DApp. In case a DApp requires the user to purchase cryptocurrency with a non-cryptocurrency, embedding payment services, such as Wyre (2019) and designing the purchasing process similar to the one of familiar platforms (C66-69) could provide for a better usability than buying a DApp-specific token on an exchange, where chart layouts can appear confusing to users (Figure 14). To counter price volatility, stablecoins have been mentioned as a recommendation during the interview process (A85, A178-179, C87-89). To save gas fees and to reduce interactions with the blockchain, one interviewee recommended to batch data on the client, to then send the data to the blockchain in a single transaction, rather than sending a transaction for each single piece of data that an action on the client produced (B147-148). As pointed out by interviewee B, gas fees can also be decreased by using stateless smart contracts (B145-146) or by storing and referencing data in the IPFS-network (B147-149). Furthermore, layer-2 technologies can be benefitial in significantly reducing gas fees and in providing a more seamless experience of DApps by increasing the speed of the settlement of transactions (A044-045, B154-163, C046). Even though this thesis specifically investigates DApps that are built using the Ethereum blockchain, other distributed ledger structures exist, such as DAG-structures, which can allow for faster transaction speeds (chapter 2.2.5). One could imagine the possibility to use other smart contract enabling distributed ledger structures as an off-chain solution in the future (as pointed out by interviewee B – B187-204) to significantly increase the transaction throughput and the transaction settlement speed, especially if Ethereum's EVM is used. To tackle the issue of the user not knowing the time until settlement of the transaction, it might be benefitial for wallet software to calculate and display an estimated timespan. Another recommendation could be to utilize the approach of Wessling et al. (2018, pp. 43-37) to assess the system design and the trust and interaction relations of all involved users of a DApp. Wessling et al. (2018, pp. 43-37) find areas of a system to be most suitable for blockchain technology, in which transitive trust relations exist and in which a need for documentation and payment processing exists. This way, elements or functionalities of a DApp could be identified, that don't benefit from being deployed on a blockchain and that could rather be deployed on a server to potentially enable faster data processing. To illustrate, an exemplary DApp architecture could therefore include both smart contracts and a server, whereby the smart contracts delegate heavy data-processing tasks to the server using oracles.

Privacy hasn’t been mentioned as an usability challenge during the interview, yet it has been considered a usability issue in chapter 3. Even though technologies, such as state channels (Coleman et al. 2018, p. 18), and services like Textile.io (Patsakis, C., & Casino, F 2019, p. 17) can increase privacy, privacy-preserving transactions are significantly cost-inefficient (Unterweger et al. 2019, p. 4), as public blockchains like Ethereum are are lacking privacy due to their transparent ledger structure (Unterweger et al. 2019, p. 4). Concerning security, DApp developers can minimize the risk of attacks by using audit services and tools (Di Angelo and Salzer 2018, p. 9) or by using secure smart contract templates to build on (OpenZeppelin 2019). Additionally, DApp developers can mitigate the risk of funds being lost due to their smart contracts being attacked through specific insurance services (Karp and Melbardis 2019). Furthermore, displaying security best practices, as Binance does (see page 104), could help a user in developing security-related habits.

The following recommendations (marked with “R”) can be synthesized and haven’t been mentioned in chapter 5.3.3:

R1 Relieving the user from specifying transactional details

R2 Automatically adjusting the UI to the desired exposure of blockchain technology and security preferences of the user

R3 Displaying a dialog that shows a summary of the intended transaction and that asks the user for confirmation

R4 Making use of unique visualizations of addresses (already mentioned as an advantage on page 60, feature by Metamask)

R5 Using Ethereum Name Service (ENS 2019) domains

R6 Avoiding blockchain terminology and rather use metaphors that are familiar to the user

R7 Automatically setting the gas price according to the speed preference of the user (feature of Metamask)

R8	Converting and displaying the amount of cryptocurrency in a familiar non-cryptocurrency (already mentioned as an advantage on page 60, feature by Metamask)
R9	Using libraries of UI-elements and DApp-components that the user is already familiar with from other DApps
R10	Enabling subscriptions for recurring payments (EIP-1337)
R11	Calculating and displaying the estimated time for transaction settlement
R12	Reassess the DApp architecture using the four-step approach proposed by Wessling et al. (2018, pp. 43-37) to find potential ways to increase transaction speed
R13	State channels and services like Textile.io can help to conserve privacy for certain use cases
R14	Smart contract security audit services and tools and secure smart contract templates can increase the security of a DApp
R15	Insurance covering smart contract attacks can mitigate risk
R16	Displaying security best practices

7 Concluding Summary and future Research Directions

With regard to the influence usability can have on the adoption of applications, this research has been conducted to provide an overview of the usability challenges DApps face and to subsequently present recommendations to overcome the identified challenges. The circumvention of intermediaries entails certain challenges and the movement towards decentralization, privacy and security can affect the usability in a negative way (A012-016). With most DApps being used in financially-related categories (Wu K. 2019, p. 3,4), applications that rely on financial incentives, such as fundraising, have seen the highest adoption of end-users (B218-B220). By merging insights from three methods, it has been revealed that financial incentivisation, value volatility and speculation can influence the usability of DApps and pose certain challenges. DApps are being built in an industry that has been created with a focus on the technology, rather than the end-user and standards (such as EIPs) are only beginning to emerge. The technical emphasis gets reinforced by the usage of blockchain specific terminology in DApps, which hinders the learning process of users unfamiliar with blockchain technology. Technical advancements are being published at a significant pace and scalability is not being seen as a problem anymore due to *layer-2* solutions that facilitate transactions beside the Ethereum main chain. This way, transactions can be settled within less than one second and more seamless experiences, with the blockchain technology fading into the background, can be enabled. Furthermore, key management and onboarding services that provide alternative ways to interact with DApps with less effort for the user, are being created. With that, barriers to using a DApp, such as having to acquire Ether first (through gas paid by the developers) or having to download a wallet software (through services like Fortmatic, 2019), are reduced. Using interactions and terminologies that users, who have no previous knowledge of blockchain technology, are familiar with, as well as hiding complex blockchain-related mechanisms, can be benefitial for the usability and might enable a higher adoption. To counter the negative influence speculation can have on the usability, stablecurrencies, that are pegged to a non-cryptocurrency, could be a solution. However, an evolutionary process of experimentation might be necessary, to explore and find products and applications of DApps that users find valuable. A developed infrastructure of tools and services that developers can build DApps on, might be able to accelerate this process.

7.1 Limitations

In this research, usability challenges and accompanying recommendations have been synthesized using limited data that has been gathered by conducting three methods. Various other usability challenges can be identified when looking at technical advancements and specific usage scenarios of DApps, for example when a participant of a state channel doesn't intend to update the main chain with the most recent state of the state channel (Coleman et al. 2018, p. 8), or when investigating specific DApp use cases other than decentralized exchanges. Additionally, usability challenges can also be user-specific: A user with more funds might put more emphasis on security than a user with less funds (A032-35). In this research, the focus has only been on a user with no previous knowledge of blockchain technology, who uses a DApp for the first time (As in chapter 4). With the DApp ecosystem and its technical advancements evolving at a significant pace, the presented list of challenges and recommendations is not exhaustive and certain challenges, which have been outlined in this research, might be resolved in the near future. Furthermore, as the industry is being viewed as "fragmented" (A110-113), with various companies tackling similar challenges, maintaining an overview of the current technical developments can be challenging. Therefore, additional recommendations could be formulated based on technical innovations or novel services that are offered, but that were unnoticed during this research. Moreover, different technical insights are being published by developers in an informal way on sites like Medium (2019), which therefore have not been addressed in the context of this research.

7.1 Future Research Directions

Due to the significant speed of technical progression the investigation of usability challenges requires ongoing effort. Therefore, a constant process of delineating the status quo of DApp usability is needed. Furthermore, the limitations of this research pose possible fields of future research: Examining the usability of DApps of categories other than "Exchanges", as well as studying the usability from the perspectives of different user types might yield valuable insights. Besides investigating the usability for the end-user, examining the usability of programming and debugging DApps for developers can also open up a different research field. Regarding the conduction of user walkthroughs, it could be benefitial for future research endeavours to let the reviewer choose applications that he is not already familiar with and to complement the cognitive walkthrough process with user tests. Another possible research direction is the investigation of ways in which off-chain interactions and the experience of transactions, that happen between layer-2 technologies and the main chain, can be designed and expressed to the user, as one interviewee pointed out the necessity to design off-chain interactions in a creative way (C049-055). A possible, different research direction is the question, whether financial incentives hinder the emergence of standards. As DApp developers put certain transactions off-chain due to lower transaction costs (C094-107), another question that might arise is, whether sidechains can have a negative influence on the adoption of the main chain.

Bibliography

ACM (2018) ACM Digital Library. Retrieved April 29th, 2018 from https://dl.acm.org/

Anceaume, E., Lajoie-Mazenc, T., Ludinard, R., & Sericola, B. (2016, October). Safety analysis of Bitcoin improvement proposals. In 2016 IEEE 15th International Symposium on Network Computing and Applications (NCA) (pp. 318-325). IEEE.

IEEE (2018) IEEE Xplore Digital Library. Retrieved April 29, 2018 from https://ieeexplore.ieee.org/Xplore/home.jsp

Alternative.me (2018). Crypto Fear and Greed Index. Retrieved May 4th, 2018 from https://alternative.me/crypto/fear-and-greed-index/

Bargas-Avila, J. A., & Hornbæk, K. (2011, May). Old wine in new bottles or novel challenges: a critical analysis of empirical studies of user experience. In Proceedings of the SIGCHI conference on human factors in computing systems (pp. 2689-2698). ACM.

Benčić, F. M., & Žarko, I. P. (2018, July). Distributed ledger technology: blockchain compared to directed acyclic graph. In 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS) (pp. 1569-1570). IEEE. Retrieved 3th March 2019 from: https://ieeexplore.ieee.org/abstract/document/8416434

Benet, J. (2014). Ipfs-content addressed, versioned, p2p file system. arXiv preprint arXiv:1407.3561. Retrieved January 18th, from: https://arxiv.org/abs/1407.3561

Bevan, N. (2009, August). What is the difference between the purpose of usability and user experience evaluation methods. In Proceedings of the Workshop UXEM (Vol. 9, pp. 1-4). Retrieved 23th March 2019 from: https://pdfs.semanticscholar.org/cba7/4036995821ca560d31bf397c695a460a63a5.pdf

Biryukov, A., Khovratovich, D., & Pustogarov, I. (2014, November). Deanonymisation of clients in Bitcoin P2P network. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (pp. 15-29). ACM.

BitInfoCharts (2019) . Retrieved 23th March 2019 from: https://bitinfocharts.com/comparison/ethereum-transactionfees.html

Blackmon, M. H., Polson, P. G., Kitajima, M., & Lewis, C. (2002, April). Cognitive walkthrough for the web. In Proceedings of the SIGCHI conference on human factors in computing systems (pp. 463-470). ACM. Retrieved 30th April 2019 from: https://dl.acm.org/citation.cfm?id=503459

Block Lattice (2019) Block Lattice. In: Nano Github. Retrieved 3th March 2019 from: https://github.com/nanocurrency/nano-node/wiki/Block-lattice

Brave (2019) Brave Browser. Retrieved 10th March 2019 from: https://brave.com/

Buterin, V., & Griffith, V. (2017). Casper the friendly finality gadget. arXiv preprint arXiv:1710.09437. Retrieved May 1st, 2019, from: https://arxiv.org/pdf/1710.09437.pdf

Buterin, V. (2014). A next-generation smart contract and decentralized application platform. white paper. Retrieved January 1st, 2019, from https://github.com/ethereum/wiki/wiki/White-Paper#decentralized-autonomous-organizations

Cai, W., Wang, Z., Ernst, J. B., Hong, Z., Feng, C., & Leung, V. C. (2018). Decentralized applications: The blockchain-empowered software system. IEEE Access, 6, 53019-53033. Retrieved 3th March 2019 from: https://ieeexplore.ieee.org/document/8466786

Chainlink (2019) Chainlink Documentation. Retrieved 3th March 2019 from: https://docs.chain.link/docs

Chandler, P., & Sweller, J. (1991). Cognitive load theory and the format of instruction. Cognition and instruction, 8(4), 293-332.

Chernyaeva, A., Shirobokov, I., & Davydov, A. (2019). Game Channels: State Channels for the Gambling Industry with Built-In PRNG. IACR Cryptology ePrint Archive, 2019, 362. Retrieved 6th June 2019 from: https://dao.casino/rnd/gamechannels.pdf

Chohan, U. W. (2018). The Problems of Cryptocurrency Thefts and Exchange Shutdowns. Available at SSRN 3131702. Retrieved 26th March 2019 from: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3131702

Cipher (2019) Cipher Browser. Retrieved 10th March 2019 from: https://www.cipherbrowser.com/

Clark, J., Van Oorschot, P. C., & Adams, C. (2007, July). Usability of anonymous web browsing: an examination of tor interfaces and deployability. In Proceedings of the 3rd symposium on Usable privacy and security (pp. 41-51). ACM. Retrieved 19th March 2019 from: https://dl.acm.org/citation.cfm?id=1280687

Coinmarketcap.com (2019) Retrieved 19th March 2019 from: https://coinmarketcap.com

Coleman, J., Horne, L., & Xuanji, L. (2018). Counterfactual: Generalized state channels. Retrieved 13th May 2019 from: https://l4.ventures/papers/statechannels.pdf

Conflux (2019) Retrieved 13th May 2019 from: http://discuss.conflux.network/

Cryptokitties (2019). Cryptokitties. Retrieved April 6th, 2019, from: https://www.cryptokitties.co/

Griffith, A (2019). Dapparat.us. Retrieved June 6th, 2019, from: https://github.com/austintgriffith/dapparatus

Dappradar (2019) Dappradar Global Charts. Retrieved April 24th, 2019 from: https://dappradar.com/charts

Decentraland (2019) Decentraland. Retrieved April 6th, 2019, from: https://decentraland.org/

Delmolino, K., Arnett, M., Kosba, A., Miller, A., & Shi, E. (2016, February). Step by step towards creating a safe smart contract: Lessons and insights from a cryptocurrency lab. In International Conference on Financial Cryptography and Data Security (pp. 79-94). Springer, Berlin, Heidelberg. Retrieved April 3th, 2019, from: https://eprint.iacr.org/2015/460.pdf

Deloitte (2018) Deloitte's 2018 global blockchain survey. Retrieved February 21th, 2019 from: https://www2.deloitte.com/content/dam/Deloitte/cz/Documents/financial-services/cz-2018-deloitte-global-blockchain-survey.pdf

De Roode, G., Ullah, I., & Havinga, P. J. (2018, December). How to break IOTA heart by replaying?. In 2018 IEEE Globecom Workshops (GC Wkshps) (pp. 1-7). IEEE. Retrieved 5th April 2019 from: https://ris.utwente.nl/ws/files/85899682/IoTA_Paper_Final_version.pdf

Dettling, W. (2018). How to teach blockchain in a business school. In Business Information Systems and Technology 4.0 (pp. 213-225). Springer, Cham. Retrieved 23th March 2019 from: https://link.springer.com/chapter/10.1007/978-3-319-74322-6_14

Di Angelo, M., & Salzer, G. (2019). A survey of tools for analyzing ethereum smart contracts. In 2019 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPCON). IEEE.

Digiconomist (2019) Bitcoin Energy Consumption Index. Retrieved May 1st, 2019, from: https://digiconomist.net/bitcoin-energy-consumption

Discord (2019) Discord. Retrieved May 1st, 2019, from: https://discordapp.com

Dziembowski, S., Faust, S., & Hostáková, K. (2018, October). General state channel networks. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (pp. 949-966). ACM.

EIP (2019). Ethereum Improvement Proposals. Retrieved June 6th, 2019, from: https://eips.ethereum.org/

EIP-20 (2018) Eip-20.md. Retrieved January 6th, 2019, from: https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md

EIP-1337 (2019) Eip-1337 – Subscriptions on the blockchain. Retrieved June 6th 2019, from: https://github.com/ethereum/EIPs/pull/1337

EIP-1613 (2019) Eip-1613.md. Retrieved June 6th 2019, from: https://github.com/ethereum/EIPs/blob/f15cd4a294791468435f1d8ae4f471cf1aab3216/EIPS/eip-1613.md

ENS (2019) Ethereum Name Service Documentation. Retrieved June 6th, 2019, from: https://docs.ens.domains

Entriken, W., Shirley, D., Evans, J., & Sachs, N. (2018). Erc-721 non-fungible token standard. Ethereum Foundation.. Retrieved January 6th, 2019, from: https://eips.ethereum.org/EIPS/eip-721

Eskandari, S., Clark, J., Barrera, D., & Stobert, E. (2018). A first look at the usability of bitcoin key management. arXiv preprint arXiv:1802.04351. Retrieved May 20th, 2019 from: https://arxiv.org/abs/1802.04351

Ethash (2018) Ethash. Retrieved January 1st, 2019, from: https://github.com/ethereum/wiki/wiki/Ethash

Etherscan (2018) Etherscan: Contracts with verified source code only. Retrieved December 30, 2018 from https://etherscan.io/contractsVerified

Etherscan Block Time (2019) Retrieved 24th March 2019 from: Ethereum Average Block Time Chart. https://etherscan.io/chart/blocktime

Fantom (2018) Fantom Whitepaper. Retrieved 7th March 2019 from: https://fantom.foundation/contents/data/2018files/10/wp_fantom_v1.6.pdf

Fenu, G., Marchesi, L., Marchesi, M., & Tonelli, R. (2018, March). The ICO phenomenon and its relationships with ethereum smart contract environment. In 2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE) (pp. 26-32). IEEE. Retrieved June 6th, 2019, from: https://ieeexplore.ieee.org/abstract/document/8327568, pp. 26-32

Fortmatic (2019) Fortmatic Documentation. Retrieved June 6th, 2019, from: https://developers.fortmatic.com/docs

Friedlmaier, M., Tumasjan, A., & Welpe, I. M. (2018, January). Disrupting industries with blockchain: The industry, venture capital funding, and regional distribution of blockchain ventures. In Venture Capital Funding, and Regional Distribution of Blockchain Ventures (September 22, 2017). Proceedings of the 51st Annual Hawaii International Conference on System Sciences (HICSS). Retrieved June 1st, 2018, from: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2854756

Gabrielli, S., Mirabella, V., Kimani, S., & Catarci, T. (2005, September). Supporting cognitive walkthrough with video data: A mobile learning evaluation study. In Proceedings of the 7th international conference on Human computer interaction with mobile devices & services (pp. 77-82). ACM.

Gartner (2018) 5 Trends Emerge in the Gartner Hype Cycle for Emerging Technologies. Retrieved April 17, 2018 from https://www.gartner.com/smarterwithgartner/5-trends-emerge-in-gartner-hype-cycle-for-emerging-technologies-2018/

Grätzer, G., Davey, B. A., Freese, R., Ganter, B., Greferath, M., Jipsen, P., ... & Wehrung, F. General Lattice Theory. 2003.

Gudgeon, L., Moreno-Sanchez, P., Roos, S., McCorry, P., & Gervais, A. (2019). SoK: Off The Chain Transactions. IACR Cryptology ePrint Archive, 2019, 360.

Hanley, B. P. (2013). The false premises and promises of Bitcoin. arXiv preprint arXiv:1312.2048. Retrieved April 17, 2018 from https://arxiv.org/abs/1312.2048

Hassenzahl, M., & Tractinsky, N. (2006). User experience-a research agenda. Behaviour & information technology, 25(2), 91-97.

Hollingsed, T., & Novick, D. G. (2007, October). Usability inspection methods after 15 years of research and practice. In Proceedings of the 25th annual ACM international conference on Design of communication (pp. 249-255). ACM.

IFPS (2019) IFPS Documentation – What is IFPS? Retrieved April 18th, from: https://docs.ipfs.io/introduction/overview/

ISO 9241 (2018) ISO 9241-11:2018 Ergonomics of human-system interaction -- Part 11: Usability: Definitions and concepts. Retrieved April 28th, 2019, from: https://www.iso.org/standard/63500.html

ISO 25010 (2019) ISO 25010:2011 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. Retrieved January 28th, 2019, from: https://www.iso.org/obp/ui/#iso:std:iso-iec:25010:ed-1:v1:en

ISO 9241-110 (2006) Ergonomics of human-system interaction — Part 110: Dialogue principles. Retrieved January 28th, 2019, from: https://www.iso.org/standard/38009.html

IOTA addresses (2019) How addresses are used in IOTA. Retrieved 5th March 2019 from: https://iotasupport.com/how-addresses-are-used-in-IOTA.shtml

Iyer, K., & Dannen, C. (2018). The ethereum development environment. In Building Games with Ethereum Smart Contracts (pp. 19-36). Apress, Berkeley, CA. Retrieved 4th April 2019 from: https://link.springer.com/chapter/10.1007/978-1-4842-3492-1_2

Jokela, T., Iivari, N., Matero, J., & Karukka, M. (2003, August). The standard of user-centered design and the standard definition of usability: analyzing ISO 13407 against ISO 9241-11. In Proceedings of the Latin American conference on Human-computer interaction (pp. 53-60). ACM.

John, B. E., & Packer, H. (1995, May). Learning and using the cognitive walkthrough method: a case study approach. In Proceedings of the SIGCHI conference on Human factors in computing systems (pp. 429-436). ACM Press/Addison-Wesley Publishing Co..

Johnson, S., Robinson, P., & Brainard, J. (2019). Sidechains and interoperability. arXiv preprint arXiv:1903.04077. Retrieved 6th June 2019 from: https://arxiv.org/abs/1903.04077

Johnston, D., Yilmaz, S. O., Kandah, J., Hashemi, F., Gross, R., & Wilkinson, S. (2015). The general theory of decentralized applications. Retrieved December 17, 2018 from https://github.com/DavidJohnstonCEO/DecentralizedApplications/blob/master/README.pdf

Kallio, H., Pietilä, A. M., Johnson, M., & Kangasniemi, M. (2016). Systematic methodological review: developing a framework for a qualitative semi-structured interview guide. Journal of advanced nursing, 72(12), 2954-2965.

Jourenko, M., Kurazumi, K., Larangeira, M., & Tanaka, K. (2019). SoK: A Taxonomy for Layer-2 Scalability Related Protocols for Cryptocurrencies. IACR Cryptology e-Print Archive, 2019, 352.

KyberDeveloper (2019) Introduction to Kyber Protocol Documentation. Retrieved 3th March 2019 from: https://developer.kyber.network/docs/Start/

Kyber Integration (2018) Decentraland's Integration with The Kyber Network. Retrieved 3th March 2019 from: https://decentraland.org/blog/announcements/decentralands-integration-with-kyber

Kyberswap.com (2019) Retrieved 19th March 2019 from: https://kyberswap.com

Law, E. L. C., Roto, V., Hassenzahl, M., Vermeeren, A. P., & Kort, J. (2009, April). Understanding, scoping and defining user experience: a survey approach. In Proceedings of the SIGCHI conference on human factors in computing systems (pp. 719-728). ACM.

Kanani J., Nailwal S., Arjun A. (2019) Matic Whitepaper. Retrieved April 22th, 2019, from: https://whitepaper.matic.network/#whitepaper

Kosba, A., Miller, A., Shi, E., Wen, Z., & Papamanthou, C. (2016, May). Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In 2016 IEEE symposium on security and privacy (SP) (pp. 839-858). IEEE. Retrieved 4th April 2019 from: https://ieeexplore.ieee.org/abstract/document/7546538

Law, E. L. C., & Van Schaik, P. (2010). Modelling user experience–An agenda for research and practice. Interacting with computers, 22(5), 313-322.

Lee J, Choi P M S (2019) Chain of Antichains: An Efficient and Secure Distributed Ledger Technology and Its Applications. Retrieved 3th April 2019 from: https://arxiv.org/abs/1810.11871

LeMahieu, C. (2018). Nano: A feeless distributed cryptocurrency network. Retrieved 3th March 2019 from: http://media.abnnewswire.net/media/cs/whitepaper/rpt/91948-whitepaper.pdf

Liu, C., Wang, D., & Wu, M. Vite: A High Performance Asynchronous Decentralized Application Platform. Retrieved 7th April 2019 from: https://icorating.com/upload/whitepaper/PAqg1hFm7yXepCV9LfsZSFPwzTuKEYdYfYky3aNo.pdf

Lombrozo E, Lau J, Wuille P (2018) Segregated Witness (Consensus layer). Retrieved 5th May 2019 from: https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki

Luu, L., Chu, D. H., Olickel, H., Saxena, P., & Hobor, A. (2016, October). Making smart contracts smarter. In Proceedings of the 2016 ACM SIGSAC conference on computer and communications security (pp. 254-269). ACM.

Macdonald, M., Liu-Thorrold, L., & Julien, R. (2017). The blockchain: a comparison of platforms and their uses beyond bitcoin. COMS4507-Adv. Computer and Network Security.

MakerDAO (2019) Retrieved 19th March 2019 from: https://makerdao.com

Mandel, T. (1997). The elements of user interface design (Vol. 20). New York: Wiley.

Maranhão, S., Seigneur, J. M., & Hu, R. (2019). Towards a Standard to Assess Blockchain & Other DLT Platforms. Retrieved 23th March 2019 from: https://archive-ouverte.unige.ch/unige:112558

Matrix (2018) Matrix Technical Whitepaper. Retrieved May 1st, 2019, from: https://github.com/MatrixAINetwork/WhitePaper/raw/master/MATRIXTechnicalWhitePaper.pdf

Metamask (2019) Metamask Github. Retrieved 10th May 2019 from: https://github.com/MetaMask

Metamask: Sending Transactions (2019) Sending Transactions. Retrieved 6th June 2019 from: https://metamask.github.io/metamask-docs/Main_Concepts/Sending_Transactions

Mohanty, D. (2018) Ethereum for Architects and Developers. P. 44-51. Apress, Berkeley, CA.

Morabito, V. (2017). Business innovation through blockchain. Cham: Springer International Publishing, p. 106-107, Retrieved May 6th, 2019, from: http://blockchainstudies.org/files/Morabito.pdf

Alonso-Ríos, D., Vázquez-García, A., Mosqueira-Rey, E., & Moret-Bonillo, V. (2009). Usability: a critical analysis and a taxonomy. International journal of human-computer interaction, 26(1), 53-74. Retrieved May 6th, 2019, from: https://www.tandfonline.com/doi/abs/10.1080/10447310903025552

MyEtherWallet (2019) MyEtherWallet Github. Retrieved 10th April 2019 from: https://github.com/MyEtherWallet

Neo (2019) NEO Whitepaper. Retrieved 7th April 2019 from: https://docs.neo.org/en-us/whitepaper.html

Karp H, Melbardis R (2019) Nexus Mutual. Retrieved 6th June 2019 from: https://www.nexusmutual.io/assets/docs/nmx_white_paperv2_3.pdf

Kounelis, I., Steri, G., Giuliani, R., Geneiatakis, D., Neisse, R., & Nai-Fovino, I. (2017, July). Fostering consumers' energy market through smart contracts. In 2017 International Conference in Energy and Sustainability in Small Developing Economies (ES2DE) (pp. 1-6). IEEE.

Medium (2019). Medium. Retrieved 7th April 2019 from: https://medium.com

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system.

Nielsen, J (1993) Usability Engineering. London, UK: Academic Press

Nielsen, J. (1994, April). Usability inspection methods. In Conference companion on Human factors in computing systems (pp. 413-414). ACM. Retrieved 14th April 2019 from: http://www.idemployee.id.tue.nl/g.w.m.rauterberg/lecturenotes/0h420/nielsen[1994].pdf

Nomura Research Institute (2016). Survey on Blockchain Technologies and Related Services FY2015 Report.. Retrieved 4th April, 2019, from http://www.meti.go.jp/english/press/2016/pdf/0531_01f.pdf

Norman D A (1983) Some Observations on Mental Models. In: Mental Models. Edited by Gentner, D. & Stevens, A. L. 1st Edition, New York, pp. 7-14

Notheisen B, Hawlitschek F, Weinhardt, C (2017) Breaking down the blockchain hype – towards a blockchain market engineering approach. In: ECIS 2017 Proceedings. Retrieved December 30, 2018 from: https://pdfs.semanticscholar.org/9f21/9d266984bcf1589c77080439f06d49465c69.pdf, pp. 1063-1080

OpenZeppelin (2019) Open Zeppelin Github. Retrieved 6th June, 2019, from: https://github.com/OpenZeppelin/openzeppelin-solidity

Oraclize (2019) Oraclize Documentation. Retrieved 6th June, 2019, from: https://docs.oraclize.it/

O'Regan, G. (2018). Human–Computer Interaction. In World of Computing (pp. 147-154). Springer, Cham.

Paas, F., Renkl, A., & Sweller, J. (2003). Cognitive load theory and instructional design: Recent developments. Educational psychologist, 38(1), 1-4.Retrieved 23th March 2019 from: https://doi.org/10.1207/S15326985EP3801_1

Patel, H., & Connolly, R. (2007). Factors Influencing Technology Adoption: A Review. In 8th International Business Information Management Conference, Dublin, Ireland. Retrieved February (Vol. 15, p. 2018). Retrieved January 28th, 2019, from: https://www.researchgate.net/publication/273140050_Factors_Influencing_Technology_Adoption_A_Review

Patsakis, C., & Casino, F. (2019). Hydras and IPFS: a decentralised playground for malware. International Journal of Information Security, 1-13. arXiv preprint arXiv:1905.11880.

Pervez, H., Muneeb, M., Irfan, M. U., & Haq, I. U. (2018, December). A Comparative Analysis of DAG-Based Blockchain Architectures. In 2018 12th International Conference on Open Source Systems and Technologies (ICOSST) (pp. 27-34). IEEE.

Pilkington, M. (2016). 11 Blockchain technology: principles and applications. Research handbook on digital transformations, 225., Retrieved January 5th, 2019, from: https://pa- pers.ssrn.com/sol3/Papers.cfm?abstract_id=2662660

Polson, P. G., Lewis, C., Rieman, J., & Wharton, C. (1992). Cognitive walkthroughs: a method for theory-based evaluation of user interfaces. International Journal of man-machine studies, 36(5), 741-773. Retrieved 14th March 2019 from: https://www.sciencedirect.com/science/article/pii/002073739290039N

Poon, J., & Buterin, V. (2017). Plasma: Scalable autonomous smart contracts. White paper, 1-47. Retrieved 6th June 2019 from: https://plasma.io/plasma.pdf

Popov, S. (2016) The tangle. Retrieved 5th March 2019 from: http://www.tangle-blog.com/wp-content/uploads/2016/11/IOTA_Whitepaper.pdf

Porru, S., Pinna, A., Marchesi, M., & Tonelli, R. (2017, May). Blockchain-oriented software engineering: challenges and new directions. In 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C) (pp. 169-171). IEEE.. Retrieved 5th April 2019 from: https://arxiv.org/abs/1702.05146

Portis (2019) Portis Documentation. Retrieved 6th June 2019 from: https://docs.portis.io/#/

Pustišek, M., & Kos, A. (2018). Approaches to front-end iot application development for the ethereum blockchain. Procedia Computer Science, 129, 410-419. Retrieved 5th March 2019 from: https://reader.elsevier.com/reader/sd/pii/S1877050918302308?token=F53BF26FBC0051C55EA31EE5AC80A2B850C3F92040088D1F3D2DB4B518CA5263D205CBF06406C296F71CD1A2C261D237

Raval, S. (2016) Decentralized Applications: Harnessing Bitcoin's Blockchain Technology. O'Reilly. ISBN 978-1-4919-2452-5

Sas, C., & Khairuddin, I. E. (2017, May). Design for trust: An exploration of the challenges and opportunities of bitcoin users. In Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (pp. 6499-6510). ACM. Retrieved April 20th, 2018 from https://dl.acm.org/citation.cfm?id=3025886

Seeger, R. (2016). Ethereum: Rise of the World Computer. Retrieved December 17, 2018 from https://www.svds.com/ethereum-the-rise-of-the-world-computer/

Solidity Introduction to Smart Contracts (2019). Introduction to Smart Contracts. In: Solidity Documentation. Retrieved January 6th, 2019, from: https://solidity.readthedocs.io/en/v0.5.2/introduction-to-smart-contracts.html

Solidity Types (2019). Solidity in Depth: Types. In: Solidity Documentation. Retrieved January 6th, 2019, from: https://solidity.readthedocs.io/en/v0.5.2/types.html

Solidity ABI (2019). Contract ABI Specification In: Solidity Documentation. Retrieved January 6th, 2019, from: https://solidity.readthedocs.io/en/v0.5.2/abi-spec.html

Sauro, J. (2015). SUPR-Q: a comprehensive measure of the quality of the website user experience. Journal of usability studies, 10(2), 68-86. Retrieved May 24th, 2019 from: http://uxpajournal.org/wp-content/uploads/sites/8/pdf/JUS_Sauro_Feb2015.pdf

Schmutz, P., Heinz, S., Métrailler, Y., & Opwis, K. (2009). Cognitive load in ecommerce applications: measurement and effects on user satisfaction. Advances in Human-Computer Interaction, 2009, 3. Retrieved 23th April 2019 from https://dl.acm.org/citation.cfm?id=1608869

Schroeder B (2003) Ordered Sets: An introduction. Retrieved 7th March 2019 from: https://www.springer.com/cda/content/document/cda_download-document/9783319297866-c1.pdf?SGWID=0-0-45-1564046-p179583643

SegWit Rebased (2019) Retrieved 5th March 2019 from: https://github.com/bitcoin/bitcoin/pull/8149

Sockin, M., & Xiong, W. (2018). A model of cryptocurrencies. Unpublished manuscript, Princeton University. Retrieved February 24th, 2019 from: https://wxiong.mycpanel.princeton.edu/papers/Crypto.pdf

Sompolinsky, Y., & Zohar, A. (2015, January). Secure high-rate transaction processing in bitcoin. In International Conference on Financial Cryptography and Data Security (pp. 507-527). Springer, Berlin, Heidelberg. Retrieved 5th March 2019 from: https://eprint.iacr.org/2013/881.pdf

Standardized Contract APIs (2015). Standardized_Contract_APIs. In: Ethereum wiki. Retrieved January 6th, 2019, from: https://github.com/ethereum/wiki/wiki/Standardized_Contract_APIs/499c882f3ec123537fc2fccd57eaa29e6032fe4a

State of the dapps (2018). Dapp statistics. Retrieved December 4th, 2018 from https://www.stateofthedapps.com/stats

Szabo N (2008) Bit gold. Retrieved January 1st, 2019, from: http://unenumerated.blogspot.com/2005/12/bit-gold.html

Swarm (2019) Swarm Documentation – 1. Introduction. Retrieved January 18th, from: https://swarm-guide.readthedocs.io/en/latest/introduction.html

Szabo N (1996) Smart Contracts: Building Blocks for Digital Markets. Retrieved January 5th, 2019, from: http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_2.html

Tasca, P. (2018) Token-Based Business Models. In: Disrupting Finance. FinTech and Strategy in the 21st Century. P. 136-148. Retrieved May 6th, 2019, from: https://link.springer.com/chapter/10.1007/978-3-030-02330-0_9

Teutsch, J., & Reitwießner, C. (2017). A scalable verification solution for blockchains. Retrieved June 6th, 2019, from: https://people. cs. uchicago. edu/teutsch/papers/truebit pdf.

Thielsch, M. T., Hassenzahl, M., & Nikolaeva, D. (2009). User Experience–aus Sicht der Forschung. Tagungsband UP09. Retrieved May 22th, 2019, from: https://dl.gi.de/handle/20.500.12116/5529

Token Tracker (2019) Etherscan Token Tracker. Retrieved May 6th, 2019, from: https://etherscan.io/tokens

Tron (2018) Tron Whitepaper. Retrieved 7th May 2019 from: https://tron.network/static/doc/white_paper_v_2_0.pdf

Twitter (2019) Twitter. Retrieved 7th May 2019 from: https://twitter.com

Unterweger, A., Knirsch, F., Leixnering, C., & Engel, D. (2018, February). Lessons Learned from Implementing a Privacy-Preserving Smart Contract in Ethereum. In 2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS) (pp. 1-5). IEEE.

Victor F, Lüders B K (2019) Measuring Ethereum-based ERC20 TokenNetworks. Retrieved June 6th, 2019, from: http://fc19.ifca.ai/preproceedings/130-preproceedings.pdf

Wessling, F., Ehmke, C., Hesenius, M., & Gruhn, V. (2018, May). How much blockchain do you need? towards a concept for building hybrid dapp architectures. In 2018 IEEE/ACM 1st International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB) (pp. 44-47). IEEE. Retrieved 11th April 2019 from: https://ieeexplore.ieee.org/document/8445058

Wessling, F., & Gruhn, V. (2018, April). Engineering Software Architectures of Blockchain-Oriented Applications. In 2018 IEEE International Conference on Software Architecture Companion (ICSA-C) (pp. 45-46). IEEE. Retrieved April 28th, 2019, from: https://ieeexplore.ieee.org/abstract/document/8432174/authors#authors

Wood, G. (2017) ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER EIP-150 REVISION. Retrieved January 6th, 2019, from: https://github.com/ethereum/yellowpaper/files/1348380/Paper.pdf

Wu, K. (2019). An Empirical Study of Blockchain-based Decentralized Applications. arXiv preprint arXiv:1902.04969. Retrieved 2th March, 2019 from: https://arxiv.org/abs/1902.04969

Wyre (2019) Wyre Documentation. Retrieved 6th June 2019 from: https://docs.sendwyre.com/docs/general

Xu, X., Weber, I., Staples, M., Zhu, L., Bosch, J., Bass, L., ... & Rimba, P. (2017, April). A taxonomy of blockchain-based systems for architecture design. In 2017 IEEE International Conference on Software Architecture (ICSA) (pp. 243-252). IEEE. Retrieved 5th April 2019 from: https://ieeexplore.ieee.org/abstract/document/7930224

Zhang, P., White, J., Schmidt, D. C., & Lenz, G. (2017). Design of blockchain-based apps using familiar software patterns to address interoperability challenges in healthcare. In PLoP-24th Conference On Pattern Languages Of Programs. Retrieved April 1st, 2019 from: https://www.dre.vanderbilt.edu/~schmidt/PDF/PLoP-2017-blockchain.pdf

Zheng, Z., Xie, S., Dai, H., Chen, X., & Wang, H. (2017, June). An overview of blockchain technology: Architecture, consensus, and future trends. In 2017 IEEE International Congress on Big Data (BigData Congress) (pp. 557-564). IEEE. Retrieved 1st May 2019 from: http://ieeexplore.ieee.org/docu- ment/8029379/, pp. 557-564

Zheng, Z., Xie, S., Dai, H. N., Chen, X., & Wang, H. (2018). Blockchain challenges and opportunities: a survey. International Journal of Web and Grid Services, 14(4), 352-375. Retrieved 1st April 2019 from: https://www.henrylab.net/wp-content/uploads/2017/10/blockchain.pdf, pp. 352-375

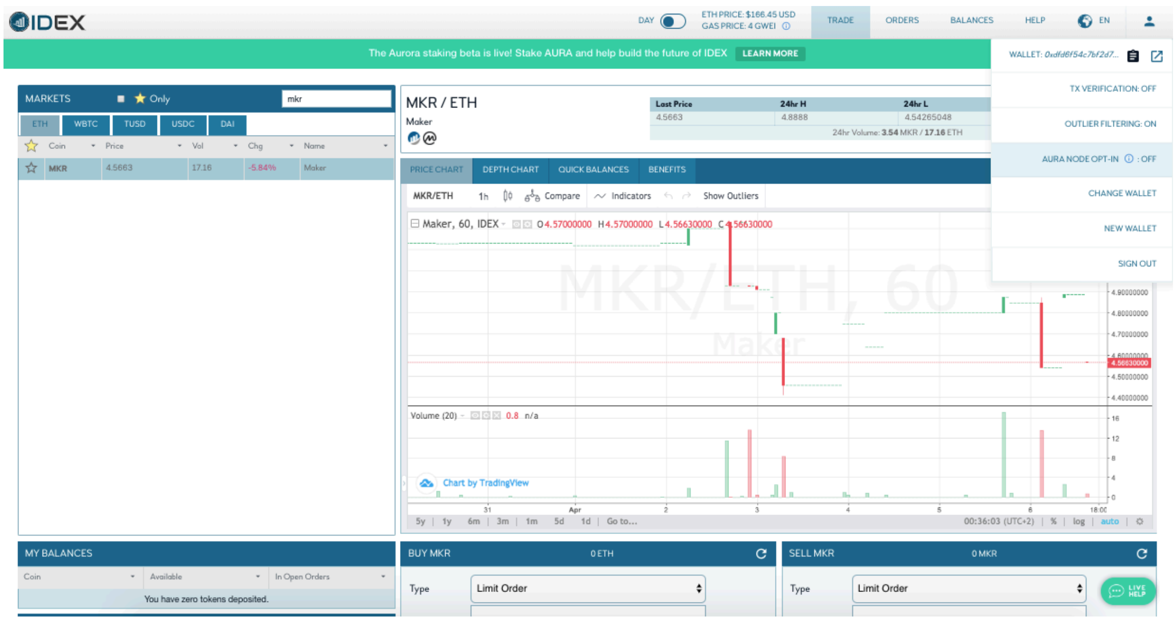
Zhou, T. (2012). Examining mobile banking user adoption from the perspectives of trust and flow experience. Information Technology and Management, 13(1), 27-37. Retrieved April 18th, 2019 from: https://link.springer.com/article/10.1007/s10799-011-0111-8, pp. 27-37

Zoom (2019) Zoom Video Conferencing Software. Retrieved June 6th, 2019 from: https://zoom.us

Appendix

A Cognitive Walkthrough: Idex

When starting the DApp by navigating to idex.market, the user gets presented a landing page that might be confusing to a novice user, who hasn't exchanged cryptocurrencies before (G7). Besides a navigation bar, the website immediately displays a chart layout, a list with all available markets and a table of the bid- and ask-orderbook. However, a modal window appears, that overlays the rest of the site and introduces the Idex exchange and provides a brief, 3-step instruction on how to start trading (G1). Furthermore, the window provides links to a site with "detailed instructions" and a site with "frequently asked questions". In case the user has questions, the window refers to a chat group and a live-chat that can be started by clicking on a floating, fixed button on the lower right area of the website. The window ends with two buttons: "Explore" and "Unlock wallet".



ETH PRICE: $166.45 USD
GAS PRICE: 4 GWEI

The Aurora staking beta is live! Stake AURA and help build the future of IDEX

WALLET: 0xdfd6f54c7bf2d7...

TX VERIFICATION: OFF
OUTLIER FILTERING: ON
AURA NODE OPT-IN : OFF

Coin	Price	Vol	Chg	Name
MKR	4.5663	17.16	-5.84%	Maker

Last Price	24hr H	24hr L
4.5663	4.8888	4.54265048

24hr Volume: **3.54** MKR / **17.16** ETH

Coin	Available	In Open Orders

You have zero tokens deposited.

BUY MKR 0 ETH
Type: Limit Order

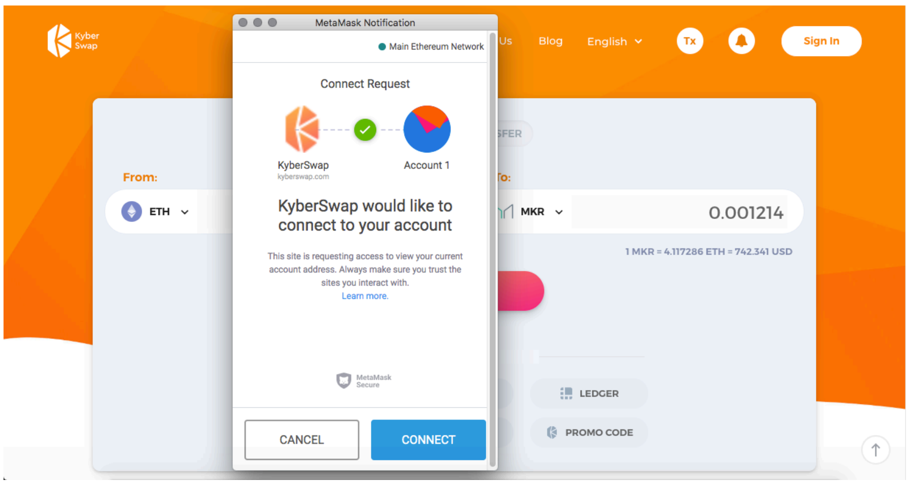
SELL MKR 0 MKR
Type: Limit Order

Figure **10**: The Idex UI with the user being logged in.

Upon clicking “unlock wallet”, a subsite presents the user four ways to unlock his wallet. Since the user hasn’t created a wallet yet, the user selects the navigationbar-button “new wallet” (**T1**). The next site displays an inputfield with the suggestion to “enter a strong password”. This step not only explains the technical background of the necessity of this action (“This password is used to encrypt your private key into a downloadable JSON file.”), it also highlights the importance of saving this password and the inability of recovering the password at a later stage (“Save this password! You cannot recover or reset it. IDEX cannot recover or reset it either”). Moreover, the direct question “Do you understand?” helps in drawing the user’s attention to the importance of saving the password (G5). Additionally, the site supports the user in ensuring a certain level of security, as a warning gets displayed in case the user enters a password that contains less than eight characters. For this walkthrough, the password gets written down on a notebook. Additionally, the private key gets saved in the browser so the user will be automatically logged in at a future visit (**T2**). Upon confirming this step with the button “Yes – create wallet”, a new site prompts the user to download his keystore file, which has been explained on the password-creation site before. Besides instructing the user how to securely save the file, the site warns the user to not use the file on any other site, indicating that his funds “can (and will) be stolen”. Even though this suggestion might make the user feel uncomfortable, it helps the user to not make a “dangerous error”, from which he might not be able to recover (G5). The following site displays the corresponding private key with the option to go back to the previous site to save his keystore file (G4). In the next step, the user unlocks his wallet, after which he gets directed to the start site with a chart layout and the orderbook. The navigationbar, as illustrated by figure 10, now displays an icon with a person and a “balances”-button, indicating the user that he is now “logged in”. When hovering over the icon, a sidebar appears, which presents the user various options, such as to track his transactions using the third-party etherscan.io site, or, to change the wallet or to sign out. However, the technical language of the presented options “Aura Node Opt-in” and “TX verification” might be confusing for a novice user (G6). Further explanation explains the staking mechanism of Idex, whereby the user learns that *AURA* is the respective cryptocurrency of the Idex DApp. Even though the user might benefit from activating the "Auro Node Opt-in"- option and also contribute to the project (for example by hosting the orderbook), it is assumable that the technical jargon and the complex system of running a PoS-node might deter the novice user from making use of this option. To achieve the first step of the goal of exchanging MKR for Ether (**T3**), the user enters the project name "Maker" in the "Markets"-input field and selects the "MKR"-ticker symbol. The site switches to a different chart with a different orderbook, showcasing the price history of the "MKR"-Ether trading pair, together with visually prominent "buy-" and "sell"-buttons. It is assumed, that the user has experience with trading financial assets, as no guidance or introduction to exchanging Ether for Maker is immediately visible to the user. A novice user might expect the "buy-" and "sell-" buttons to exchange the assets within one click, however, clicking the "buy"-button displays a dialog that tells the user to "enter a valid buy order" (G2). As the user intends to exchange digital assets of value, it is assumable, that the user might be afraid of making an error and therefore doesn't want to try out the interface without having a certain understanding of the functionality (G7). Furthermore, the field in the "buy-" section indicates that the user's has 0 Ether to trade, even though the user has Ether in his wallet, which might be confusing to the user. At this point, the user might be unaware, that he first has to deposit Ether to the smart contract of the DApp, before being able to trade (G1). A link in the footer of the site directs the user to a visually comprehensive guide on how to trade on idex. Having read the guide, the user navigates to "Balances", where he chooses to deposit Ether to the Idex smart contract. Besides specifying the amount of Ether the user intends to send, he also has to choose the amount of gas he wants to include. A novice user might be unfamiliar with the concept of gas and might leave the gas-specification at the default value (5 gwei) (G6). Furthermore, no information is provided that converts the amount of Ether the user intends to deposit to US-dollars, which the user might be more familiar with. A second modal window appears, that asks the user for confirmation of the value the user wants to send (G5). Upon confirming the deposit action, no dialog is being presented to the user that provides information about the success or status of the deposit (G3). When refreshing the site, the Ether value is deducted from the wallet from the user, however, the “My Idex Balance”-value amounts to 0. After a minute, in which the user might be concerned with his deposit being lost, the correct deposited value appears in the “My Idex Balance”. The user heads back to the exchange interface and chooses between the options of creating a “market-“ and “limit”-order in the “buy“-section. The site provides a wide array of information about the supply and demand of the market, such as a price graph including various charting tools, a market depth chart and an orderbook, which might overwhelm a novice user (G7). By typing in the amount of Ether the user has available to trade in the “Total”- input field, the lowest “ask”-price of the sellers gets filled in and the resulting “MKR”-value gets calculated, whereby the total equivalent value in US-dollars doesn’t get displayed. Upon clicking the “buy”-button, a modal dialog window confirms the successful trade (T3). Moreover, his updated MKR-balance gets displayed in the “sell“-field (**T4**). Even though the user buys the cryptocurrency he wants, he might buy it at an unfavourable rate, as certain trading pairs have a significant price difference between the lowest sell- and the highest buy-order, which might not be clear for a novice user. To withdraw the MKR-currency from the smart contract to his wallet, he navigates to “Balances” and clicks “withdraw”, after which the MKR-balance of his wallet gets updated after a certain delay (**T4**). To “log out”, the user opens the sidebar and chooses the “Log out”-button (**T5**). A modal window asks the user for confirmation, which might save the user from losing his funds in case he forgot to write down his private key (G4). Afterwards, another modal window signals the user that he has “successfully logged out” (G3).

B Cognitive Walkthrough: Kyberswap and Metamask

The user begins by opening kyberswap.com. The landing page doesn’t display any chart layout, but presents the user a visually prominent container, which gives the user the option to “swap” and “transfer” tokens. Inside the centered container, two input fields are provided, which lets the user select and type in any type of token he wishes to swap. It is assumable, that a novice user can understand the functionality and next step required to exchange a cryptocurrency, based on the layout and presentation provided (G1). Overall, the language is familiar to a novice user, the token value the user types in gets automatically converted to US-dollar and no technical terms are used (G6). Six different ways are offered for the user to connect his wallet, however, for this walkthrough, the Metamask browser plugin is used, as this way of accessing the wallet is also mentioned as part of the “self-confirmed transactions”-pattern (Wessling and Gruhm 2018, p. 46). Upon typing in the desired amount in Ether and clicking the “Metamask”-button, a modal window with an error message, suggesting the user to install “Metamask”-first, gets presented (G4). However, no link to download the Metamask-plugin is provided, so the user has to manually search for a site to download Metamask himself. Installing the Metamask-plugin can be perceived as a straightforward process, as the plugin gets added to the browser within a few seconds and the user gets redirected to a “welcome“-page, that guides the user through the process of creating a new wallet. Yet it is to note that the user can’t import an Ethereum account using a private key, as a seed phrase generated by Metamask is required to import an existing account. That way, a user new to Metamask would have to send funds to the newly created wallet before interacting with the DApp using Metamask. The process of creating a new wallet (**T1**) through Metamask is succinct and bundled into sequential steps. Furthermore, instead of confronting the user with hashes, the user has to choose a password, which a novice user might be more familiar with. Afterwards, a “secret backup phrase” consisting of 12 words gets generated, that helps the user to recover his account. The presentation of this step is seemingly targeted at a novice user: Familiar language is used (“secret backup phrase” instead of “mnemonic seedphrase”, G6) and tips and suggestions to save the phrase are given (“Write this phrase on a piece of paper and store in a secure location.”, G1). Additionally, the phrase is blurred out at first and the user has to type in the phrase in the next step for further security (G5). The user consequently writes down his login information on a notebook (**T2**). After completing the wallet-creation process, the user is immediately logged in and can use the plugin. The user doesn’t need to save his private key or public key, as they get saved by Metamask (**T2**).

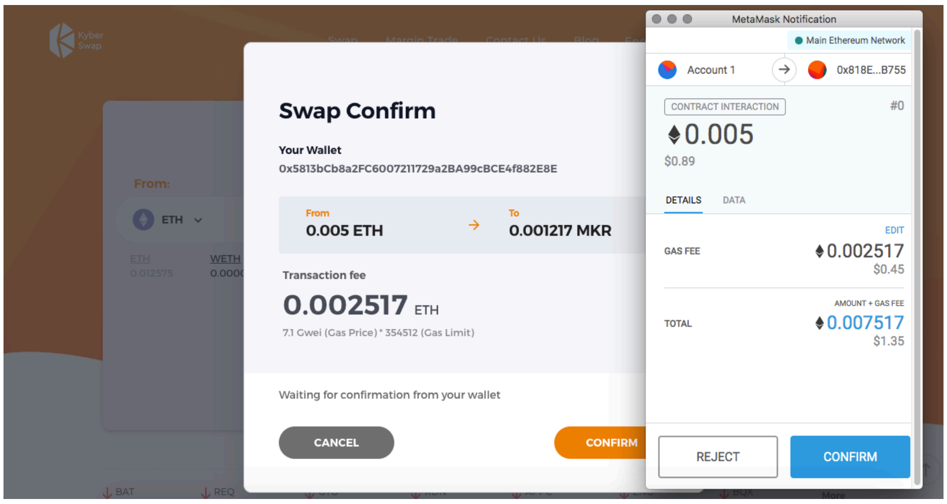


KyberSwap would like to connect to your account

This site is requesting access to view your current account address. Always make sure you trust the sites you interact with.
Learn more.

Figure **11**: The Kyberswap UI with a Metamask popup window

When switching back to kyberswap, the same error window appears upon clicking the “Metamask”-button, which can be confusing to the user (G1), so the user has to reload the site. After the user clicks the “Metamask”-button again, a popup-window appears, which asks for the user’s permission to connect with the DApp (Figure 11). This might create a feeling of control and security for the user. Additionally, each public key is associated with a unique colored icon that Metamask displays, which helps the user in visually seperating and identifying accounts. After allowing the DApp to connect with Metamask, a modal window displays the word “processing” for less than three seconds, however, the expected transaction doesn’t get initiated and no feedback gets provided. The user assumes, that a technical error has happened and checks his wallet balance to make sure that his funds didn’t change. The user returns to the website and clicks on a "swap"-button, after which a modal window gets presented, that lists the transaction details, including the gas fee and requests the confirmation of the user (**T3**). Upon clicking "confirm", a Metamask popup appears, that also asks the user for confirmation of the transaction (figure 12). Furthermore, both gas fee and transaction value are converted to US-dollar.


Swap Confirm

Your Wallet
0x5813bCb8a2FC6007211729a2BA99cBCE4f882E8E

From 0.005 ETH → To 0.001217 MKR

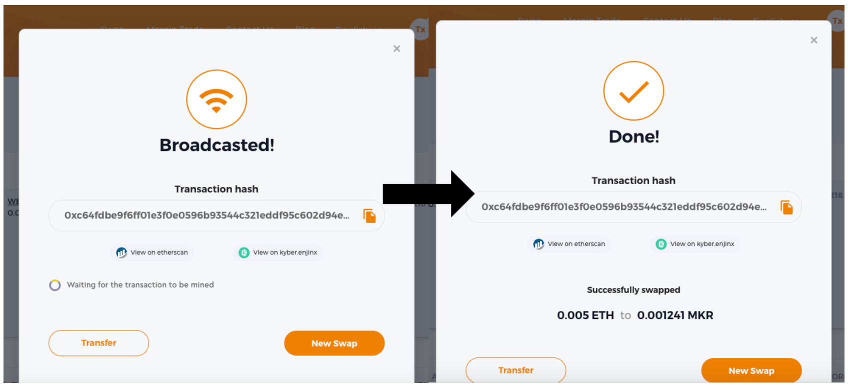
Transaction fee
0.002517 ETH
7.1 Gwei (Gas Price) * 354512 (Gas Limit)

Waiting for confirmation from your wallet

MetaMask Notification
Main Ethereum Network
Account 1 → 0x818E...B755
CONTRACT INTERACTION #0
0.005
$0.89
GAS FEE 0.002517 $0.45
TOTAL (AMOUNT + GAS FEE) 0.007517 $1.35

Figure **12**: The Kyberswap UI with a Metamask popup window

After the user confirms the transaction, a modal window displays the information, such as the transaction-hash, and the status of the transaction (G8 – figure 13). After the transaction the user can access all historical transactional information at any time using his Metamask plugin (**T4**). As the user hasn't "logged in" in the DApp, but rather allowed the connection between Metamask, which controls his account, and kyberswap, he can leave the site without having to sign out (**T5**).


Broadcasted!

Transaction hash
0xc64fdbe9f6ff01e3f0e0596b93544c321eddf95c602d94e...

Waiting for the transaction to be mined

Done!

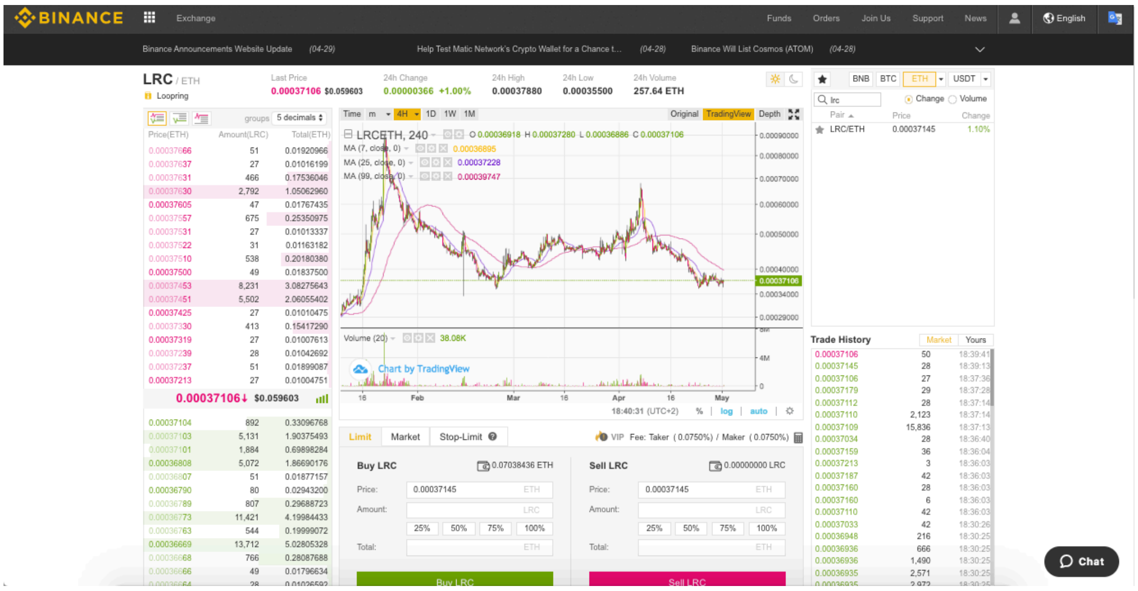
Transaction hash
0xc64fdbe9f6ff01e3f0e0596b93544c321eddf95c602d94e...

Successfully swapped
0.005 ETH to 0.001241 MKR

Figure **13**: The Kyberswap UI displaying the status of a transaction

C Cognitive Walkthrough: Binance

The cryptocurrency exchange Binance, which is mentioned as an example of the pattern "delegated transactions" (Wessling and Gruhn 2018, pp. 45,46), features a landing page that presents a slider with announcements and a list of markets, that are available to trade. A salient "Create Account"-button directs the user to a registration page, which requires the user to input his email-address, his preferred password, an optional referral-id and his consent to the terms of use. Furthermore, a "support"-button is provided in the footer, which opens a third-party help desk website in case the user has questions (G4). Upon filling out the information, a modal window displays a verification in form of a puzzle, that aims at checking whether the user is human. Having sent the registration form, a new site prompts the user to check his email account for a confirmation email. Due to many web services using this form of registration, a novice user might be used to create a new account this way (G7). Like with other web applications, the user either chooses to let Chrome automatically save his email-password combination or he writes down his account credentials on a notebook (**T2**). The confirmation email contains security suggestions (G5) and a link to an endpoint of the Binance website, which navigates the user to a page stating that the user has successfully created his account (**T1,** G3). A "Log in"-button leads the user to a login-page. After having successfully logged in by filling out an email- and passwordfield and by completing another puzzle verification, the user gets redirected to a "safety risk notice"-site, which contains a checklist of security suggestions (G5), such as never telling account-relevant information to anyone.or making sure to visit the correct page. Only after having checked all checkboxes the user is able to click a button labeled "I understand, continue", which opens a site, that provides an overview of the user's account information. Within less than three seconds, a modal dialog appears, that displays a warning with the suggestion to the user to enable a two-factor authentication for further security (G5) by either using a SMS-authentication or installing an authentication app. After choosing to skip this type of verification, the user gets access to his account page, which offers the user a variety of functionalities regarding the management of his account, such as changing his password, setting up an “anti-phising-code” or labeling and managing withdrawal addresses for enhanced clarity and security. Moreover, this page features a list of all devices that accessed the user’s account, including their operation system, location and IP and time of logging in (G8). This can create a reassuring feeling for the user concerning his account’s security. A switch with the label “Use BNB to pay for fees” indicates that the user has to pay for each transaction, however, the user might not know at this stage that “BNB” is the respective cryptocurrency of the Binance exchange. To achieve the goal of exchanging Ether for LRC (**T3**), the user hovers over the “Exchange”-button in the navigationbar, which opens a drop-down menu with the options “Basic” and “Advanced”. Having had no previous experience with exchanging digital assets before, the user chooses the “Basic”-option, which leads the user to a page with a chart layout, an orderbook, a list with all available markets and the options to buy and sell cryptocurrencies using “limit-“, “market-“ and “stop-limit”-orders (Figure 14).



Binance Announcements Website Update (04-29) Help Test Matic Network's Crypto Wallet for a Chance t... (04-28) Binance Will List Cosmos (ATOM) (04-28)

LRC / ETH Loopring

LRC / ETH	Last Price	24h Change	24h High	24h Low	24h Volume
Loopring	0.00037106 $0.059603	0.00000366 +1.00%	0.00037880	0.00035500	257.64 ETH

groups 5 decimals

Price(ETH)	Amount(LRC)	Total(ETH)
0.00037666	51	0.01920966
0.00037637	27	0.01016199
0.00037631	466	0.17536046
0.00037630	2,792	1.05062960
0.00037605	47	0.01767435
0.00037557	675	0.25350975
0.00037531	27	0.01013337
0.00037522	31	0.01163182
0.00037510	538	0.20180380
0.00037500	49	0.01837500
0.00037453	8,231	3.08275643
0.00037451	5,502	2.06055402
0.00037425	27	0.01010475
0.00037330	413	0.15417290
0.00037319	27	0.01007613
0.00037239	28	0.01042692
0.00037237	51	0.01899087
0.00037213	27	0.01004751
0.00037106↓	$0.059603	
0.00037104	892	0.33096768
0.00037103	5,131	1.90375493
0.00037101	1,884	0.69898284
0.00036808	5,072	1.86690176
0.00036807	51	0.01877157
0.00036790	80	0.02943200
0.00036789	807	0.29688723
0.00036773	11,421	4.19984433
0.00036763	544	0.19999072
0.00036669	13,712	5.02805328
0.00036668	766	0.28087688
0.00036666	49	0.01796634
0.00036664	28	0.01026592

Limit | Market | Stop-Limit

VIP Fee: Taker (0.0750%) / Maker (0.0750%)

Buy LRC	0.07038436 ETH	Sell LRC	0.00000000 LRC
Price:	0.00037145 ETH	Price:	0.00037145 ETH
Amount:	LRC	Amount:	LRC
	25% 50% 75% 100%		25% 50% 75% 100%
Total:	ETH	Total:	ETH
Buy LRC		Sell LRC	

BNB | BTC | ETH | USDT

lrc Change Volume

Pair	Price	Change
LRC/ETH	0.00037145	1.10%

Trade History Market | Yours

0.00037106	50	18:39:41
0.00037145	28	18:39:13
0.00037106	27	18:37:36
0.00037179	29	18:37:28
0.00037112	28	18:37:14
0.00037110	2,123	18:37:14
0.00037109	15,836	18:37:13
0.00037034	28	18:36:40
0.00037159	36	18:36:04
0.00037213	3	18:36:03
0.00037187	42	18:36:03
0.00037160	28	18:36:03
0.00037160	6	18:36:03
0.00037110	42	18:36:03
0.00037033	42	18:30:26
0.00036948	216	18:30:25
0.00036936	666	18:30:25
0.00036936	1,490	18:30:25
0.00036935	2,571	18:30:25
0.00036935	2,972	18:30:25

Chat

Figure **14**: The Binance UI displaying the “Basic” chart layout

Like with the idex-exchange, the multitude of available market-information might be confusing and overwhelming for the user (G2), who might feel incompetent, considering he chose the "Basic"-option (G1). After typing in his desired ticker symbol (LRC), the chart presents the historic price information of "Loopring" and the user types in the amount of Ether he wishes to spend on the transaction. The option to calculate the respective amount of owned Ether using buttons with percentages (25%, 50%, 75% and 100%) might be helpful for the user. However, the US-Dollar value of the transaction doesn't get displayed, so the user might have to calculate the amount of Ether he intends to transact with a value of 5 US-Dollars himself. After typing in the amount of Ether for 5 US-Dollars (0.32), the amount of LRC (85), based on the price of the last transaction, gets displayed and the user clicks on the "Buy"-button. Afterwards, a notification at the top right corner appears, informing the user about the successful placement of the order (G3). However, a novice user might be unaware that his order at the bidding price might not get filled immediately and might think that he directly made the purchase (G3). Only after another participant is willing to sell at the entered price, the transaction gets facilitated. Besides being able to confirm the successful purchase by seeing the updated LRC-balance on the trading screen (G3), he can additionally go to the "Balances"-page, where the user can view an overview of his funds, calculated in Bitcoin and US-Dollar value (**T4**). To log out, the user hovers over the "Person"-icon and chooses "Logout" (**T5**).

D Evaluation Framework with a List of all identified Challenges and Advantages

Evaluation Framework (EF) (Table 5), based on Alonso-Ríos et al. (2009, p. 3 ff.), **with a list of both advantages (a) and challenges (c),** derived from technical idiosyncracies (TI), cognitive walkthroughs (CW) and semi-structured interviews (SI). Challenges and advantages that meet multiple criterias are marked with an (r).

Table 5: Evaluation Framework with a list of advantages and challenges

EF 1.	Knowability (Learnability)		
	EF 1.1 Clarity	TI	Blockchain can be perceived as a topic that is not easy to comprehend (c)
		TI	Highly specialized technical language can be confusing to novice users (C021) (c)
		TI	Metaphors, e.g. “wallet” can support the learning process (a)
		TI	The actual speed a transaction takes to be settled can only be estimated in advance (c)
		TI	Selection of a network can be confusing for a novice user (c)
		TI	Constitution of the price of the service he pays for with the DApp can be unclear to the user (c)
		TI	Transparency of all transactions and balances (r)
		CW	The presentation of financial instruments, the purchasing methods and the chart layout can be confusing to a novice user when obtaining Ether or a token (c)
		CW	Continuously informing the user about the status of a transaction (Kyberswap & Metamask) (a)
		CW	Displaying pictograms that are unique to its associated public keys (Metamask) (a)(r)
		CW	Metamask’s feature of showing the transactional costs converted to the respective US-dollar amount (a)
		SI1	Focus on security, privacy and decentralization can hinder adoption – abstraction is necessary when hiding the blockchain technology from the user (c) (A013)
		SI2	Open source development of products can lead to complicated user experiences (A124)

	EF 1.2 Consistency	TI Token prices are subject to the market and therefore (in most cases) not pegged to a certain value – transactional costs can be hard to predict (c); Price volatility also Mentioned in SI (A085) CW Reference to third-party websites (e.g. "Etherscan") for transactional information can disrupt the UX (c) SI A multitude of different ways of interacting with the blockchain exists and no clear standard exists yet – the user has to learn different user experiences, which hinders adoption (A106) (c) (r)
	EF 1.3 Memorability	TI Usage of cryptographic concepts, such as hashes (c) CW Giving the user the option to recover his account by using a sequence of words ("mnemonic phrase" - Metamask) (a)(r) CW Displaying pictograms that are unique to its associated public keys (Metamask) (a)(r)
	EF 1.4 Helpfulness	TI No central support service exists that can help the user in revoking payments or getting his private keys back (c) TI Few resources exist that users can turn to in case they need help (c)
EF 2.	Operability (Effectiveness)	
	EF 2.1 Completeness	SI The industry's approach to build technologies first without having a clear view of the target user (A020) (c) (r)
	EF 2.2 Precision	
	EF 2.3 Universality	TI Hardware requirements can exclude certain users from participating in the consensus process (c)
	EF 2.4 Flexibility	TI Client software (such as geth) allows the user to choose from different node options (a) CW Metamask's feature of allowing the user to export the private key, thus letting the user control his exposure to blockchain technology (a) CW Metamask's feature of requesting the approval each time a transaction gets initiated (a)(r) SI The industry's approach to build technologies first without having a clear view of the target user (A020) (c) (r)

EF 3.	Efficiency	
	EF 3.1 Efficiency in human effort	TI Need to acquire Ether before interacting with a DApp (c)(r) TI Some DApps require the user to purchase their respective token beforehand (c) TI Effort when calculating the equivalent value in US-dollars a user might be more familiar with (c) TI Significant effort and resources required when participating in the consensus process and running a node (c)(r) CW Email-&passwordcombination provides the highest convenience to gain control over an account, yet it requires the highest amount of trust (a) CW Metamask's feature of staying "logged in" when interacting with DApps (a) CW Setting default values (e.g. gas) (a)(r) CW Giving the user the option to automatically calculate a certain percentile of the owned Ether amount (Binance)(a) SI Acquiring cryptocurrencies with non-cryptocurrencies is considered difficult (A070) (c) SI A multitude of different ways of interacting with the blockchain exists and no clear standard exists yet – the user has to learn different user experiences, which hinders adoption (A106) (c) (r) SI Effort to download & install software like Metamask can result in significant bounce rates (B005) (c) SI Software like "Metamask" requires knowledge about blockchain technology due to blockchain terminologies and security practices (C021) (c)
	EF 3.2 Efficiency in task execution time	TI Transactions with a higher gasprice get processed faster (a) TI Congestions in times of higher usage can lead to higher confirmation times and higher average gas fees (c)(r) TI Full Nodes require significant time to synchronize with the blockchain (c) CW Transaction delays: Caused Idex to not display the sent amount on both the smart contract account and the imported wallet account for a brief moment, which might make the user believe that his funds are lost (c)

		SI Most current blockchain-based games don't utilize layer 2 technologies, thus the gameplay experience is unlike non-blockchain games (C047)
	EF 3.3 Efficiency in tied up resources	TI Congestions in times of higher usage can lead to higher confirmation times and higher average gas fees (c)(r) TI Significant effort and resources required when participating in the consensus process and running a node (c)(r)
	EF 3.4 Efficiency in economic costs	TI Necessity to pay a certain fee for every transaction can make certain applications (e.g. IoT) inefficient (c) TI Congestions in times of higher usage can lead to higher confirmation times and higher average gas fees (c)(r) SI Using the Ethereum's blockchain for single transactions can be significantly cost- inefficient (B145) (c)
EF 4.	Robustness (Error Tolerance)	
	EF 4.1 Robustness to internal error	TI The call of a smart contract fails in case the specified gas limit is lower than the actual gas needed for execution (c)(r) TI Node synchronization with the blockchain can be highly unreliable (c) TI Light node found to be error-prone - can cause certain events to be lost (c)
	EF 4.2 Robustness to improper use	TI Users can't recover from potential errors they make – also applies to the IPFS-network (c) TI A transaction with a gasprice that is significantly lower than the average gasprice might not be processed (c) CW Security suggestions and guidelines can prevent the user from making critical errors. (a)(r) CW Setting default values (e.g. gas) (a)(r)
	EF 4.3 Robustness to third party abuse	TI Network attacks can potentially change the integrity of the ledger (c) TI Transaction fees can prevent nodes from spamming transactions (a) TI Current techniques of encryption might be ineffective in the future (c) TI Censorship-resistance (a)

	EF 4.4 Robustness to environment problems	TI The outage of one node doesn't affect the availability of the transaction records (a)
EF 5.	Safety – the "capacity to avoid risk and damage"	
	EF 5.1 User safety	TI Typing errors when directly specifying the ABI can lead to the user's funds to be lost (c)(r) TI The call of a smart contract fails in case the specified gas limit is lower than the actual gas needed for execution and the user loses his gas fee (c)(r) TI Unlike passwords, private keys can't be recovered (c) TI User has to be aware of possible attacks such as through malware, phishing or theft (c) TI Certain methods can potentially reveal the identity of an agent within the network network (c) CW Security suggestions and guidelines can prevent the user from making critical errors. (a)(r) CW Giving the user the option to recover his account by using a sequence of words ("mnemonic phrase" - Metamask) (a)(r) CW Metamask's feature of requesting the approval each time a transaction gets initiated (a)(r) CW Setting default values (e.g. gas) (a)(r)
	5.2 Third party safety	TI Network attacks can declare previously valid transactions as valid (c)
	5.3 Environment safety	
EF 6.	Satisfaction - extended with selected UX factors identified by Bargas-Avila and Hornbaek (2011, p. 2693)	
	EF 6.1 Enjoyment	
	EF 6.2 Motivation	TI Transaction fees incentivize and motivate miners in PoW blockchains (a) TI Transparency of all transactions and balances might discourage users who intent to hide sensitive information (c)(r)

		TI User might view the token as an investment vehicle and therefore might be reluctant to use the token in exchange for services the DApp provides (c); also mentioned in SI (A178) SI Convincing value to the user outside of pure speculation purposes is considered challenging (A073, B019) (c)
	EF 6.3 Frustration	TI Typing errors when directly specifying the ABI can lead to the user's funds to be lost (c)(r) TI Users who are used to immediate responses from non-DApps can be frustrated from transaction delays (c) TI The call of a smart contract fails in case the specified gas limit is lower than the actual gas needed for execution and the user loses his gas fee (c)(r) TI Need to acquire Ether before interacting with a DApp (c)(r)
	EF 6.4 Trust	TI The risk of a ledger to be attacked could lead to distrust and a reluctance to use a DApp (c) TI Irreversibility as a "risk"-factor when dealing with dishonest transaction partners (c) TI User has to trust in the integrity and security of the external wallet provider TI The transparency of being able to track transactions increases the trust a user has in the system (a) TI Regarding exchanges: The historic and expected future performance, along with the reputation, the trading activity and the support service contribute to the user's trust CW Different architectures of DApps exist, which feature a certain trade-off between convenience and trust (a) SI Distrust in stablecurrencies (A085)

E Evaluation Framework with a List of all identified Recommendations

Evaluation Framework (EF) (Table 6), based on Alonso-Ríos et al. (2009, p. 3 ff.), **with a list of recommendations,** that got explicitly mentioned in the semi-structured interviews (SI) in chapter 5.3.3 and recommendations, that got synthesized in chapter 6 (SYN). Recommendations that meet multiple criterias are marked with an (r).

Table 6: Evaluation Framework with a list of all identified recommendations

EF 1.	Knowability (Learnability)		
	EF 1.1 Clarity	SI	Allowing a paradigm shift towards abstraction to happen (for a better understanding for the user), even at the potential expense of decentralization, pricacy and security (A059-060, A197-198, C113, CH1)
		SI	Using authentication and key management solutions that are familiar to a novice user and reducing effort by not forcing the user to download additional software like Metamask, for example using a email & password combination (Portis 2019) or the phone number (Fortmatic 2019) (B118-119, C019-029, A042) (r)
		SI	Designing creative ways and user experiences when interacting with sidechains (C049-055)
		SI	Develop protocols and tools in an "open source"-way, but not products and user experiences, to prevent "complicated" experiences (A124). (CH9)
		SYN	Making use of unique visualizations of addresses (already mentioned as an advantage on page 60) (r)
		SYN	Displaying a dialog that shows a summary of the intended transaction and that asks the user for confirmation (r)
		SYN	Using Ethereum Name Service (ENS 2019) domains (r)
		SYN	Avoiding blockchain terminology and rather use metaphors that are familiar to the user (r)
		SYN	Converting and displaying the amount of cryptocurrency in a familiar non-cryp- tocurrency (already mentioned as an advantage on page 60)
		SYN	Using libraries of UI-elements and DApp-components that the user is already familiar with from other DApps (r)

		SYN Calculating and displaying the estimated time for transaction settlement
	EF 1.2 Consistency	SI Promote collaboration among the industry to help the evolvement of industry- wide standards of interacting with the blockchain (A112, CH8) SI Development of industry-wide standards of user experience, technology, terminology and interaction with the blockchain (A130-132, A147) SI Utilizing stablecoins, such as DAI, to mitigate the issue of volatility and the possible reluctance to spend tokens on DApps (A173-183, C087-089, C119-121) (r) SI Using libraries of UI-elements and DApp-components that the user is already familiar with from other DApps (r)
	EF 1.3 Memorability	SYN Making use of unique visualizations of addresses (already mentioned as an advantage on page 60) (r) SYN Using Ethereum Name Service (ENS 2019) domains (r)
	EF 1.4 Helpfulness	
EF 2.	Operability (Effectiveness)	
	EF 2.1 Completeness	
	EF 2.2 Precision	
	EF 2.3 Universality	
	EF 2.4 Flexibility	SI Building DApps and web3 products while having a clear understanding of the targeted end-user (A033, CH2) SI Deploying a smart contract using generalized state channels increased flexibility and cost-efficiency (B122) (r) SI Different onboarding experiences in regards to target users (B138-139) (r) SYN Automatically adjusting the UI to the desired exposure of blockchain technology and security preferences of the user SYN Automatically setting the gas price according to the speed preference of the user (feature of Metamask) (r)
EF 3.	Efficiency	

	EF 3.1 Efficiency in human effort	SI Gradually exposing the user to the underlying blockchain technology (A152-162), as the "stake of the user increases" (A152-162) (r) SI Using authentication and key management solutions that are familiar to a novice user and reducing effort by not forcing the user to download additional software like Metamask, for example using a email & password combination (Portis 2019) or the phone number (Fortmatic 2019) (B118-119, C019-029, A042) (r) SI Paying the gas fees on behalf of the users (B121, C114) (r) SI Implementing non-cryptocurrency to cryptocurrency payment services due to familiarity and convenience (C067-068) SYN Relieving the user from specifying transactional details SYN Automatically setting the gas price according to the speed preference of the user (feature of Metamask) (r) SYN Using Ethereum Name Service (ENS 2019) domains (r) SYN Using libraries of UI-elements and DApp-components that the user is already familiar with from other DApps (r) SYN Enabling subscriptions for recurring payments (EIP-1337)
	EF 3.2 Efficiency in task execution time	SI Using layer-2 technology to accelerate the transaction speed, instead of facilitating all transactions on the main chain (A044-045, C046, CH11, CH13), using state channels (B154-155) or sidechains, such as Matic Network (C031-036, B159-163, B174) (r) SI R1 Reassess the DApp architecture using the four-step approach proposed by Wessling et al. (2018, pp. 43-37) to find potential ways to increase transaction speed
	EF 3.3 Efficiency in tied up resources	SI Use IPFS to store large amounts of data outside of the smart contract (B147)
	EF 3.4 Efficiency in economic costs	SI Using layer-2 technology to accelerate the transaction speed, instead of facilitating all transactions on the main chain (A044-045, C046, CH11, CH13), using state channels (B154-155) or sidechains, such as Matic Network (C031-036, B159-163, B174) (r) SI Paying the gas fees on behalf of the users (B121, C114) (r) SI Deploying a smart contract using generalized state channels increased flexibility and cost-efficiency (B122) (r) SI Utilize stateless smart contracts for decreased gas costs, so data can get accessed from the messages, which maintain the state, rather than the contract (B145-146) SI Batching transactions to reduce gas fees (B147-148) (r)

EF 4.	Robustness (Error Tolerance)	
	EF 4.1 Robustness to internal error	
	EF 4.2 Robustness to improper use	
	EF 4.3 Robustness to third party abuse	
	EF 4.4 Robustness to environment problems	
EF 5.	Safety – the "capacity to avoid risk and damage"	
	EF 5.1 User safety	SI Hardware key encryption can improve the user's security of his funds not being stolen (A214-226) SYN Displaying a dialog that shows a summary of the intended transaction and that asks the user for confirmation (r) SYN State channels and services like Textile.io can help to conserve privacy for certain use cases SYN Smart contract security audit services and tools and secure smart contract templates can increase the security of a DApp SYN Insurance covering smart contract attacks can mitigate risk SYN Displaying security best practices
	5.2 Third party safety	
	5.3 Environment safety	
EF 6.	Satisfaction - extended with selected UX factors identified by Bargas-Avila and Hornbaek (2011, p. 2693)	
	EF 6.1 Enjoyment	

	EF 6.2 Motivation	SI Conveying value to the user that's outside of speculation (A072-073) SI Utilizing stablecoins, such as DAI, to mitigate the issue of volatility and the possible reluctance to spend tokens on DApps (A173-183, C087-089, C119-121) (r)
	EF 6.3 Frustration	SI Gradually exposing the user to the underlying blockchain technology (A152-162), as the "stake of the user increases" (A152-162) (r) SYN Displaying a dialog that shows a summary of the intended transaction and that asks the user for confirmation (r) SYN Avoiding blockchain terminology and rather use metaphors that are familiar to the user (r)
	EF 6.4 Trust	SI Foster trust in stablecurrencies (A086)