Weber, Ingo; Staples, Mark

Programmable money: next-generation blockchain-based conditional payments

Digital Finance

Provided in Cooperation with:
Springer Nature

Suggested Citation: Weber, Ingo; Staples, Mark (2022) : Programmable money: next-generation blockchain-based conditional payments, Digital Finance, ISSN 2524-6186, Springer International Publishing, Cham, Vol. 4, Iss. 2, pp. 109-125, https://doi.org/10.1007/s42521-022-00059-5

Digital Finance (2022) 4:109–125
https://doi.org/10.1007/s42521-022-00059-5

ORIGINAL ARTICLE

Programmable money: next-generation blockchain-based conditional payments

Ingo Weber[1] · **Mark Staples**[2,3]

Received: 5 May 2021 / Accepted: 26 July 2022 / Published online: 2 September 2022

Abstract

Conditional payments allow the transfer of money only when pre-defined rules hold. Example uses could include welfare payments, employee expenses, insurance payouts, or tied donations. Normally, conditions are checked manually in reimbursement or pre-approval/audit processes, either at accounts before funds are distributed, or using account records after distribution. Blockchain's capabilities for smart contract and digital assets can be used to implement next-generation conditional payments, on a decentralized ledger. We conducted a project with an industry partner where we conceptualized, implemented, and evaluated a novel programmable money concept using blockchain. In the system, programmed policies are not attached to accounts, but instead to money itself. Policies here specify the conditions when money may be spent, and can be automatically checked by the money as it is spent. Policies can also define auxiliary actions to be taken as part of payment transactions, including side-payments. Policies can be dynamically added to and removed from money as it flows through an economy. These policies could be budget rules for tied funds, but could also enable new forms of "values-based money" that respect ethical or other rules which relate to societal values or social norms. We report on some of our experiences and insights regarding blockchain architecture, software engineering with blockchain, and blockchain-based programmable money. We also identify opportunities and open research questions in these areas.

Keywords Blockchain · DLT · Smart money · Programmable money · Conditional payment

✉ Ingo Weber
Ingo.Weber@tu-berlin.de

Mark Staples
Mark.Staples@data61.csiro.au

[1] Chair of Software and Business Engineering, Technische Universitaet Berlin, Sekr. EN-6, Einsteinufer 17, 10587 Berlin, Germany

[2] CSIRO's Data61, Sydney, Australia

[3] School of Computer Science & Engineering, UNSW, Sydney, Australia

JEL Classification E40 · E52 · I38 · O30

1 Introduction

Disruptive technologies break assumptions about limitations of systems and business models, creating new software-architectural options and challenges for solution development. Blockchain[1] breaks assumptions that a high-integrity register of digital assets must be centrally administered by a trusted party. However, blockchain also has a variety of technical challenges for their development and performance in operation, some of which are characteristic of only some blockchain platform technologies, whereas others are in part inherently unavoidable for decentralized shared ledgers. Software engineering for blockchain-based systems must adapt to respective new architectural and development challenges (Xu et al., 2019).

1.1 Digital and dentralized money

First-generation blockchains were conceived as a means to create and manage digital currency. Bitcoin demonstrated a solution to the problem of privately issued digital cash, and in particular a system that can prevent double-spending of assets, using a single *logically centralized ledger* of cryptocurrency transactions. However, despite being logically centralized, the ledger is operated in an *organizationally decentralized and physically distributed* way by a collective of thousands of nodes. The basic advantage of "digital cash" is that it can be held directly by owners like normal cash (and is not, like bank money, held by a third party that must be trusted) and that it can be transferred remotely to counterparties like bank money (and when transferred does not, like normal cash, have to be transferred in person to a co-located counterparty). Depending on how the digital money is managed and on which type of ledger, e.g., anonymous or pseudonymous, transparency of flows of money may increase.[2]

On the role of cryptocurrency, Burda (2021) writes: "The phenomenon of private currency issue goes to the root of the circularity of money's definition and its multifarious, competing attributes described 150 years ago by Jevons (1875); to some degree, they are at the same time a medium of exchange (MOE), a unit of account (UOA), a store of value (SOV), and a standard of deferred payment (SDP)." Cryptocurrencies like Bitcoin and Ether (based on the Ethereum blockchain) are highly volatile, e.g., "market valuation for private digital assets [was] at yearend 2020 at $800b, by mid-April more than $2t" (Burda, 2021). A broader analysis of volatility of Bitcoin valuation has been conducted by Schilling and Uhlig (2019a). This limits

[1] In this article, we say 'blockchain' as a short-hand for blockchain and other distributed ledger technologies.

[2] Research related to the issue of transparency is, among others, conducted by WG1 "Transparency in FinTech" of the COST Action 19130 Fintech and AI in Finance, see https://fin-ai.eu/wg1-transparency-in-fintech/, accessed 2021-06-25.

the use as UOA, SOV, and SDP, given that changes in valuation of 5–10% from 1 day to the next are not an exception, and have not been for years (Harvey, 2014).

As a somewhat ironic counterpoint, the use as MOE can be hampered by popularity: first- and second-generation blockchains are limited in their maximum throughput, and selection of transactions to be included and processes is to some degree driven by transaction fees offered by the senders of the transactions. Thus, the more overloaded a blockchain network becomes, the higher the fees climb. At the time of writing, the daily median fee per transaction for Bitcoin and Ethereum fluctuate between $2 and $12, reaching more than $30 on some days.[3] At such fees, the MOE function is mostly used for transferring more substantial values, e.g., around $1k as a median for Bitcoin.[4]

The above-mentioned limitations can be circumvented or mitigated by

(i) issuing so-called *stable coins*, which aim to avoid fluctuation in value relative to fiat currencies by some means; and
(ii) using blockchain systems and platforms with higher throughput.

Harvey (2016) compares and contrasts traditional and cryptocurrency-based means of payment in terms of risks and costs. Schilling and Uhlig investigate in which circumstances cryptocurrencies might be more suitable for some transactions than fiat moneys (Schilling & Uhlig, 2019b). Lin et al. investigate whether technical features of blockchains may influence prices and valuations (Lin et al., 2021). Furthermore, Petukhina et al. discuss the role cryptocurrencies can play in investment portfolios (Petukhina et al., 2021). Finally, Harvey, Ramachandran, and Santoro comment on decentralized finance, and the future of finance in general and the role blockchain-based methods may play (Harvey et al., 2020).

Modern blockchains support many kinds of data and digital assets, and allow transactions to record small programs ("smart contracts") and their execution. Smart contracts allow developers to create bespoke high-integrity abstractions. *Tokens* are an example—digital assets created using smart contracts, implementing high-level interfaces for digital asset transfer, but which can have highly customized behaviour. The applications and the potential benefits of blockchain applications are manifold, and blockchain has been tried out or applied in almost all industries (Bratanova et al., 2019).

1.2 Programmable money and context for this work

In collaboration with a large bank, we conducted the "Smart Money" project, to conceptualize, implement, and evaluate systems for blockchain-based *programmable money* (Royal at al., 2018). This is a novel concept where policies can be dynamically associated with parcels of money, and then be checked, updated, or removed as that

[3] See e.g. https://bitinfocharts.com/comparison/bitcoin-median_transaction_fee.html and https://bitinfocharts.com/comparison/ethereum-median_transaction_fee.html, both accessed 2021-06-11.
[4] https://bitinfocharts.com/comparison/bitcoin-mediantransactionvalue.html, accessed 2021-06-11

money is transferred. The policies attached to the money can decide whether the money is allowed to be spent in a given transaction, can initiate auxiliary actions including side-payments, and can remove themselves from the money in specific circumstances. In this paper, we use the term 'policies' to refer to these conditions or actions on parcels of money, but they are intended to be used as partial implementations of economic policies, financial control policies in business, or budget rules applied to specific monies allocated to individual economic actors.

The project was motivated by challenges observed in Australia's National Disability Insurance Scheme (NDIS), a large sophisticated system providing government support to people with disabilities. Each NDIS participant has specific funding conditions tied to individual budget lines, based on a tailored plan of supports. An example policy in our approach may thus specify that money from a budget line for physiotherapy can only be spent for such services, offered by registered physiotherapists. In the project, we showed that programmable money can be implemented well by utilising blockchain capabilities, first and foremost smart contracts.

In the literature, the term programmable money has, among others, been used to differentiate blockchain-based money (including Bitcoin) from other forms of digital money, like database tables in a banking system "holding" money digitally (Majuri, 2019). Using the terms programmable money or *smart money* to refer to money which checks that given conditions are met before it can be spent is a recent phenomenon, and the literature on that topic is as yet thin. A short discussion paper by Avital et al. (2017) summarized the key advantages of smart money, which can control when, where, and by whom it is spent, to whom it is paid, and for what. This has also been articulated in a presentation by Hedman (2019). In the above-mentioned collaborative project, we explored the concept of smart money in the context of welfare payments to disabled people (Royal at al., 2018). The usefulness of smart money in this context has been confirmed in a study by Rodrigues and Cardoso (2019). Another case study (Kolehmainen et al., 2021) aims at "digitalizing and automating processes in enterprise legacy systems" and includes conditional payment, building on our earlier work. A special case of programmable money could comprise programmable donations, as explored in an interview study (Elsden et al., 2019b). Central banks have begun to investigate the potential benefits of programmable money for Central Bank Digital Currencies, and have considered various solution concepts and their design trade-offs (BoE, 2020; BIS, 2020). As such, the research on the concept has been very limited, and with this paper, we aim to provide a spark for more work in this interesting and promising direction.

This paper reports on highlights from the "Smart Money" project (Royal at al., 2018), and expands on some of the software engineering challenges and research opportunities arising from the project. To this end, we first present the conceptual solution in Sect. 2. Then, we discuss implementation concerns in Sect. 3 and summarize the evaluation in Sect. 4. Finally, lessons learnt are described in Sect. 5 before the paper is concluded in Sect. 6. An earlier, shorter version of this paper has also been published as a non-refereed keynote paper (Weber and Staples, 2021).

2 Conceptual solution

Conditional payments are important and common. Welfare, insurance, grants, donations (Elsden et al., 2019b), and corporate expenses are all examples. Conditions are usually checked manually, after payments are made. Some existing schemes use conventional technology to dynamically check payment conditions, but only allow payments from centrally controlled accounts. In contrast, our project studied blockchain-based conditional payments with end user-controlled accounts, by devising a form of smart contract-backed programmable money. In the following, we describe the architecture and the realization of the core functionality in smart contracts.

2.1 Architecture

The smart money project investigated a new concept of blockchain-based programmable money. Rather than checking fixed conditions on payments from controlled accounts, policies for programmable money can instead be dynamically attached to and detached from money that flows through a payments system. These policies can check conditions for payment, and can also implement auxiliary actions including side-payments. Previous authors use the term "programmable money" for single-use conditions attached to cryptocurrency. Tokens on blockchains may carry reusable conditions and can also be used as digital assets (perhaps as "money"), but, normally, these conditions are fixed when the tokens are issued. In contrast, our concept of programmable money comprises not only the checking of conditions specified in flexible policies, but also that (i) new policies can be attached to money by its owner, (ii) policies by default remain attached to money as it is paid, or (iii) policies can update themselves or remove/detach themselves from money during execution. To our knowledge, this concept of programmable money is novel. The desired novel features created a host of technical challenges, among others:

- How to ensure that relevant information is present on the blockchain at the time when needed to evaluate a policy?
- How to best attach the dynamic policies to the money?
- How to enable delegation, such that a nominee of an NDIS participant can spend tokens on behalf of the latter?
- How to handle agreements for recurring payments, such as regular physiotherapy treatment?

To provide the desired features, blockchain technology with smart contract capabilities forms a natural base. Alternative, centralized realizations without blockchain as a base can be considered, and have been in the project's analytical evaluation (Royal at al., 2018). For a single-use case, e.g., a system only addressing NDIS in isolation, they may offer similar benefits. However, once more than one use case should be implemented, blockchain is the more suitable technological basis for the following reasons: (i) there would often not be a single authority (e.g., one government agency in charge) for the multiple use cases, (ii) end users would not need

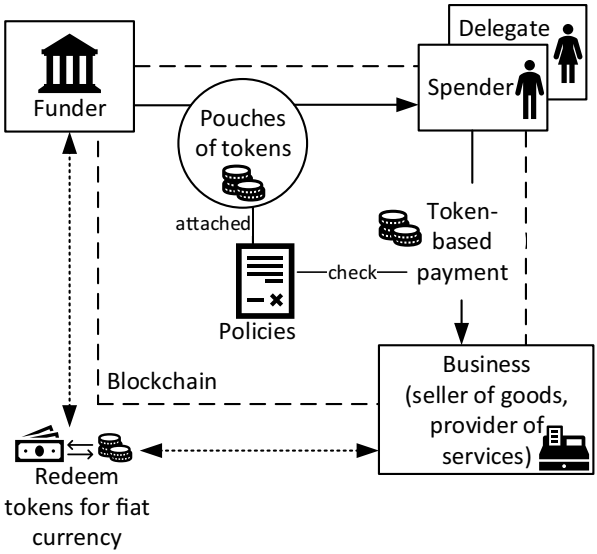


Fig. 1 Solution overview, showing policies attached to money (pouches of tokens), transferable between parties, and optionally redeemable

```
function transfer(parcel, recipient, service, amount)
  require([...]); //initial checks: spender authorized? Amount available?
  newPolicies[] = [];
  foreach (policy in parcel.policies)
    if (!policy.check(sender, recipient, service, amount))
      return false; //if the policy forbids the transfer, terminate
    else
      newPolicies.append(policy.getNextPolicies(sender, recipient, service, amount));
  parcel.subtract(amount); //remove amount from old parcel
  newParcel = Parcel(recipient, newPolicies, amount); // new parcel with new policies
```

Listing 1: Pseudo code of policy-aware transfer function

multiple accounts and multiple different processes to interact with the different use cases on different systems, and (iii) cross-links between the use cases could be implemented with relative ease on a joint platform. This discussion is continued in Sect. 5, including a concrete example. Whether or not blockchain is the best choice should always be evaluated critically, on a case-by-case basis, see e.g. Xu et al. (2019, Chapter 6), but not neglect a big-picture view such as the above-mentioned realization of multiple use cases.

An overview of the chosen on-chain architecture is depicted in Fig. 1. Payments are done on-chain using underlying tokens, where, e.g., 1 token is worth $1. To efficiently attach policies to tokens, the tokens are grouped into *pouches*, or *parcels*, and the funder (e.g., a government agency in our use case) distributes pouches to eligible spenders (e.g., NDIS participants). How the tokens are managed in pouches with attached policies is the subject of the next subsection.

Spenders might delegate rights (e.g., to carers or family members). Spenders or delegates then pay for services or goods with the tokens. The attempted payment only completes if all the policies are followed. Recipients might then spend or redeem the tokens, i.e., trade the tokens for a corresponding amount of fiat currency, to be transferred to their conventional bank accounts. Whether or not tokens can be redeemed of course depends on the policies attached to the tokens, which shows the need for dynamic attachment and removal of policies.

The on-chain architecture is complemented with a blockchain trigger as per Weber et al. (2016), i.e., a component that translates between on-chain and off-chain invocations, as well as front-end components to serve the different types of users with suitable, mobile-friendly UIs (see also Sect. 3). More details on all components can be found in Royal et al. (2018, Chapter 5). Next, we focus on the core of the back end, i.e., the on-chain realization of programmable money with smart contracts.

2.2 Realization in smart contracts

In the chosen design, one smart contract defines the framework logic for programmable money. This contract holds all parcels of money, and each parcel is subject to a (possibly empty) set of policies. The policies are also implemented by smart contracts which are either incorporated into or referenced by the smart contract representing the parcels of money. When money from a parcel is spent, the logic for funds transfer is executed as illustrated in the pseudo-code in Listing 1 . In summary, the transfer only succeeds if funds are available, the spender is authorized (line 2), and all attached policies are fulfilled (lines 4–6). This often includes a check whether the recipient is allowed to receive funds for a specific purpose, e.g., only registered physiotherapists can receive payments for physiotherapy. Each policy specifies what happens after funds are transferred ("getNextPolicies", line 8), and the "next" policies are attached to the resulting parcel of money, created in line 10 with the recipient as owner. Policies may also be removed during this step (not shown in this simplified pseudo-code). Finally, the funds are subtracted from the original parcel (line 9). As such, the original parcel is split into two with each transfer. Parcels with the same policies and owners could be merged subsequently.

An alternative solution would be to manage the tokens in higher level groups, where all tokens in such a group are subject to the same policies, and one would store the ownership of these tokens. In such a case, each group of tokens could implement a (slightly extended) ERC-20 contract interface or similar. However, this alternative would have the downside that changing the policies attached to a token would require transferring said token from one token registry to another. This would be harder to manage and verify, and would exceed the scope of token standards (like ERC-20) for the most central operation, i.e., token transfer.

Generally speaking, whenever tokens are used as money, they should only be accepted if they can be used or redeemed. In our solution, policies update themselves to facilitate this. For instance, imagine a physiotherapist is paid with money that can only be spent on physiotherapy services. If the policies remained unchanged, the

physiotherapist could themselves only use that money for physiotherapy. In our solution, such a policy would remove itself when paid to a physiotherapist.

Finally, the challenge of handling recurrent payments is addressed with "service agreement contracts" as follows. Upon agreeing on recurring services and respective payments, like regular physiotherapy treatments, a corresponding amount of tokens is calculated based on the frequency, price per treatment, and the maximum number of treatments, and this amount is provisionally flagged for payment to the chosen provider. As the services are delivered over time, payments are enabled and executed accordingly.

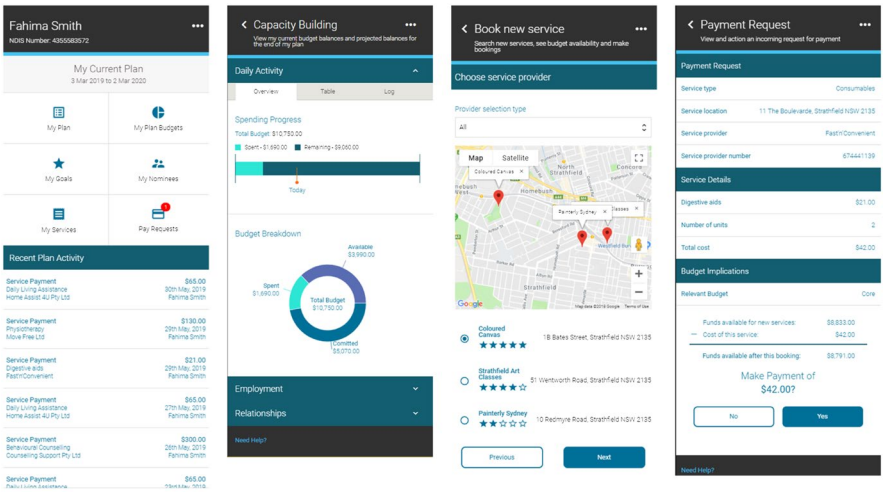
2.3 Context of the conceptual solution

Programmable money enables powerful forms of conditional payments. Dynamically modifiable policies provide a high degree of flexibility. Conditions can be about the service or asset paid for, a previous payment, or the status of another account or money (thereby also comprising a form of *higher order money*). Conditions might refer to time, place, or reliable data obtained from the ledger or as an input. When the actions are performed on programmable money, the associated policies might invoke other actions, such as making auxiliary payments (including self-taxing money), sending notifications, or triggering other business processes (Weber et al., 2016). Furthermore, programmable money can also be designed to enable strong forms of data analytics, including comprehensive access to accurate real-time data, while respecting suitable confidentiality requirements.

If implemented in practice, our programmable money concept would likely run concurrently with other existing payment mechanisms. While payments could be made between parties solely using programmable money tokens, approaches are required to allow interoperation across payment mechanisms. For example, if an NDIS service provider only accepts cash payments, then an NDIS participant paying in cash might submit their programmable money for reimbursement to the funding body, rather than using it in direct payment to the service provider. The payment conditions or underlying infrastructure must be designed to accommodate these kinds of interoperability requirements. The payment conditions might still be automatically checked at time of reimbursement, but until the time of reimbursement, there will be some exposure for the participant to the risk that the conditions will not be satisfied, and that their request for reimbursement might be declined. In the system currently in use, this reimbursement risk is faced by NDIS participants, but could be addressed by the end-to-end use of programmable money.

3 Implementation and test deployment

We implemented the concept in a working prototype. The research organization (CSIRO) developed the back-end and blockchain architecture, and the bank developed the front-end domain-specific user interfaces. The prototype was used for end-user testing and achieved a high degree of usability (Royal at al., 2018).



(a) Overview screen (b) Budget overview (c) Booking a service (d) Payment request

Fig. 2 Screenshots of the UI. Source: Royal at al. (2018); © CSIRO, reprinted with permission

Selected screenshots of the prototype's UI are shown in Fig. 2. Specifically, the overview screen for NDIS participants (Fig. 2a) lists the main features, including treatment plan, budgets, goals, nominees, services, and payment requests. On the latter, the red circle with a "1" indicates one new payment request. Below the main features follows a list of recent activities. The budget view (Fig. 2b) shows the spent and remaining amounts, as well as any previously committed portions, e.g., for recurring service accounts as described above. When booking a new service (Fig. 2c), a map and a list view show available providers, including their star rating. Payment requests (Fig. 2d) list the details of the request and allow the user to authorize the payment; doing so creates a signed blockchain transaction, which in turn leads to the evaluation of the policies attached to the parcels of money, and the payment is only successful if the policy checks evaluate to `true`.

In the following, we summarize the back-end implementation and deployment in our research project. Given the developer community, maturity, and prior experience of the project team, we decided to use the *Ethereum* blockchain platform. The system used an Ethereum deployment on a private proof-of-authority (PoA) network, where consensus was established on the authority of three known and trusted nodes. Note that, in practical deployments, users like NDIS participants would not be required to operate a node. Much like cryptocurrency holders on public blockchains like Bitcoin and Ethereum can use the so-called wallet software, e.g., on their mobile devices, an NDIS participant could use a human–computer interface for an application that manages the participant's key pair. Choices such as which consensus algorithm to use, and who runs a node, are questions outside of the scope of our project, which focused on investigating the functionality and usability of our new programmable money concept. For any real-world use of programmable money, these additional questions would need to be

answered, as they affect important properties like privacy, performance, and scalability of the resulting system. To this end, all steps of an architecture design process like the one in Xu et al. (2019, [Chapter6]) should be addressed for practical deployments.

The money was implemented as special kind of token smart contract (about 300 lines of code). This catered for dynamically establishing pouches of tokens and attaching budget policies as additional smart contracts (base policy contract: about 200 lines of code), as described in the previous section.

For our use case, NDIS, we pre-defined a set of (parametrized) policies, which were used for the different budget types and purposes of NDIS. Automated approvals were only given for registered businesses providing specific services or goods matching a participant's budget policies. To enable policy checks by the money, we implemented a registry of businesses in another smart contract. Spenders and their delegates in the project interacted with the blockchain through a mobile app, which also managed the key pairs for authorization. Integration with the front end was achieved through the joint definition of a REST API, specified using Swagger.

4 Evaluation summary

The research goal was to investigate the concept and usability of programmable money, so we accepted limitations in scope to not give complete treatments for confidentiality, performance, and reliability. We conducted various end-user, analytical, and technical evaluations; full details can be found in Royal et al. (2018, Chapter 7). Most of the nine end users (all NDIS participants or carers) who participated in the respective part of the study were highly positive about the features. This positive regard is expressed among others in a net promoter score of 89%. One aspect particularly valued by the end users is the ability to make direct payments with certainty that the policies were fulfilled. For the existing centralized NDIS system used in production, payment success rates are reported publicly[5] and the most recent report at the time of writing covers the week ending Sunday 07 February 2021, with a 96.7% success rate of payment requests out of approx. 1.3M requests; precisely, 44,377 payment requests were unsuccessful in that week. More than half of those failures were because the claimed amount was greater than was available. The desire of NIDS participants to have immediate payment certainty is hence very understandable.

In the analytical evaluation, we compared our blockchain design with three alternatives, one with currency-on-blockchain and two based on conventional technology (current and latest). Confidentiality and privacy were partly treated by having separate pseudonymous identifiers for each budget line, where the mobile app integrated those identities for each participant.

[5] https://www.ndis.gov.au/about-us/publications, see "weekly payment summaries"; accessed 2021-02-20.

As for the technical evaluation, we considered among others throughput performance. Based on the respective analysis, we believe that the system could be made to scale to adequate levels of throughput for the NDIS use case across Australia. The starting point of this analysis was an estimation from the NDIS of an average load of 2.17 payments per second for July 2020. Based on latest actual numbers (see Footnote 5), the average load was about 2.2 payments per second. In our solution, some payments would require more than one transaction, so we estimated that on average a payment results in approx. 1.5 blockchain transactions, and therefore an average throughput of 3,3 blockchain transactions per second. Even though peak demand will be higher, this could likely be handled by a deployment like the public Ethereum network (if NDIS were the only application): the average number of transactions has been above 10 since mid-2020. Achieving higher throughput in a private deployment is not hard in our experience.

Finally, we conducted additional technical tests, not all described in the project report. These included negative tests—tests supposed to fail—such as attempting to submit transactions to the blockchain directly that violated the policies. These tests resulted in exceptions as intended, demonstrating that policy integrity is enforced by the blockchain. Therefore, alternative user interfaces could be developed without impacting integrity.

5 Lessons learnt and new opportunities

We report on learnings about programmable money on blockchain as well as new opportunities arising from it, and about software engineering for blockchain-based systems.

5.1 Programmable money on blockchain

The project report (Royal at al., 2018) identified potential benefits for the funding agency, service providers, and participants. These included improved control by participants over funds, reduced risk of misuse of funds, better visibility of budget status, and improved data analytics.

However, from the users' perspective, a solution for a single funder using a blockchain would have little difference compared to one using the conventional technology. Although uncommon today, centralized databases can support user-programmable bank accounts (Elsden et al., 2019a). Users interacted with our system using web or mobile interfaces, so back-end technologies are hidden.

Nonetheless, a blockchain-based system could better integrate multiple funders. Consider a veteran with a disability and other medical conditions. They may want to access combined funds from separate agencies, each having different funding policies. Blockchain-based infrastructure could readily enable this integration, while funders retain strong levels of control.

The concept requires more than programming the money itself. The policies run as smart contracts, and so can only use information on the ledger or input during payment. In the project, we included a service provider registry on the ledger, so that the policies could check whether a payment to a given provider was allowed.

A key challenge for users is to know what policies are attached to money. Some policies might be universal, imposed by regulators (e.g., blacklisting sanctioned recipients), but others might be temporary and user-defined. Policies written in a Turing-complete programming language could have arbitrary and undecidable behaviours, or be inscrutable bytecode. Self-modifying policies and interactions between policies increase this complexity. There are many possible approaches to dealing with these challenges. Valid policies might be restricted to regulators or their delegates, or to a registry of "safe" policies. Integrity-checking functions could wrap risky policies, formal verification might be used (Magazzeni et al., 2017), and declarative smart-contract languages may make analysis easier. There should also be valid human-readable explanations of the policies.

These policy-related challenges also have economic and financial implications. When money has conditions attached, it is less powerful as a medium of exchange, and so therefore will have less value. For example, gift cards which can only be used at specific shops, trade in secondary markets at a value 5–25% lower than their face value, depending on the shop (Offenberg, 2007). Therefore, a concern with potential macroeconomic implications is that programmable money might not be worth as much as normal money. However, in our example where the conditions are spending rules set by government for the social security payments, the rules already apply, even with conventional money. The difference programmable money brings is more efficient and higher quality compliance with the existing rules.

A possible mode of application for the concept is the creation of "values-based money" in an economy. We here refer to societal values, such as sustainability, ecological goals, organic production of foodstuff, or fair trade. For example, imagine if a consumer wants to buy organic fair-trade coffee from a supermarket, and while paying for coffee, encumbers their payment with conditions that the money can only be spent within a certified organic fair-trade supply chain. The rules of such a system might allow retailers and distributors to take a capped but reasonable margin according to the conditions of the certifying body. Using the mechanisms described in this paper and our report, as payments are made to these certified intermediaries and to the final producer, the encumbering rules could be automatically removed to allow the received monies to be spent as normal money. This concept is the dual of the normal blockchain concept of ensuring high-quality certified supply chain by tracking the flow of goods; instead what is managed is the flow of money in the other direction. If values-based money only implements existing legislated conditions such as for sanctions, counter terrorism-financing, or anti-modern slavery, then programmable money would in principle only bring more efficient and higher quality compliance with those existing laws. However, if it was allowed for consumers or businesses to create or use other discretionary schemes to constrain operations on parcels of money, to support goals such as fair trade, then this might significantly complicate economic activity and potentially reduce the aggregate value of money in the economy in the same way that gift cards do. Our conception of programmable

money (and its potential use for value-based money) does not necessarily give rise to multitude permanently distinct private currencies (Schilling & Uhlig, 2019a), because spending constraints can be automatically removed from the money when their policies are fulfilled in transfer.

On the other hand, programmable money has additional functionality and potential benefits, and this may increase its value. Of course, there are basic benefits of digital cash of being both held directly like normal case, and being able to be transferred to remote counterparties, like bank money. Programmable money has additional capability such as triggered side-payments and escrow. An open question concerns the extent to which these capabilities might potentially increase the value of programmable money, compared to normal money.

5.2 Software engineering for blockchain systems

Blockchains are almost always components in larger systems incorporating conventional technologies, including for key management, user interfaces, communication, and systems' integration. As a component, a blockchain functions as a database, a communication mechanism, a computational platform, and often as a mechanism for asset control and management [Xu et al. (2019, Chapter5)]. Software engineering with blockchain is similar in many ways to conventional technologies, but there are differences.

Some aspects of software engineering with blockchain are similar:

- Usability and user experience remain critical. Our project demonstrated through end-user testing that blockchain-based systems can be highly usable, despite programmable money's conceptual and architectural complexity.
- In our system, the blockchain was invisible to users, and we believe that this will be typical for many blockchain-based systems.
- Front-end development is largely unaffected by blockchain, because user interfaces often access blockchains through normal APIs.
- Clear integration APIs and testing are critical to combine architectural components, including the blockchain. We used conventional REST APIs written in Swagger for this purpose.

Some aspects are different:

- Blockchain ledgers are immutable, but this is where smart contracts are deployed and executed. Facilities to enable updates of smart contracts must be provided for in the design.
- Architectural decisions about allocation of functions to components remain critical, but for blockchains, this includes deciding what data and functions should be on-chain (Xu et al., 2019). In particular, moving business logic on-chain might allow process redesign (Mendling et al., 2018).
- How to horizontally scale components that create and submit transactions on behalf of a single party is not necessarily trivial: an Ethereum account is associ-

ated with a key pair, which can be supplied to multiple machines, but each transaction has a unique number, the *nonce*. This nonce can be thought of as a transaction sequence number for a given account, and transactions are processed in the order of the nonces and only when all previous nonces have been processed. Each nonce may of course only be used once. This creates synchronization concerns if multiple machines should create transactions originating from a single account. While the transactionality should be resolved by the consensus mechanism, it may complicate users' experience.

- In other work BIS (2020), authors have been concerned about scalability in respect of the computational cost of smart contracts "on ledger", and have suggested that moving this computation to independent modules or significantly restricting its functionality may be required to achieve full scalability. In our view, these on-ledger costs are unlikely to have a significant unresolvable impact to scalability of throughput or latency, but it is an open question.
- For smart contracts to check policies, data must be on-chain or supplied during invocations; typically, smart contracts cannot directly invoke external APIs. must be either provided during invocation, or previously be recorded on the ledger. This impacts the design of data storage and component interactions.
- Similarly, test data must also be on-chain, and re-running tests requires re-establishing the test environment. In our project, this was achieved by re-creating the entire blockchain for each test run. Now-available emulation tools like Ganache can be of help.
- Key management is critical for authorization in blockchain systems, and this is more complex and risk-prone architecturally and for users than centralized credential management.
- Blockchain ledgers are transparent, so the operating collective can cross-check ledger integrity. This makes confidentiality hard to achieve. Design tactics to use pseudoanonymity and encryption may not stop linking attacks. Yet, regulators or courts may require access to "private" data; and consumers often value transparency. These trade-offs have also been recognised by others (BIS, 2020). Resolving these issues is a significant design challenge in blockchain projects.

6 Conclusions, open research questions, and outlook

Blockchain gives software engineers new options when developing systems to solve user problems, but also brings new challenges for software engineering practice like those listed in the previous section and in Sect. 2. We have described some of our experiences and learnings from our "Smart Money" project (Royal at al., 2018), which developed and tested a novel concept of programmable money on blockchain.

In our concept, policies can be dynamically attached to and removed from pouches of money. These can be checked when payments are made, and can stay attached to the money as it flows through a payment system. Rather than just checking conditions, the policies can also perform actions such as making auxiliary payments, or updating or removing themselves. This concept could not be readily

implemented using conventional technologies, but is reasonably straightforward to implement using smart-contract tokens on a blockchain.

Although the concept seems complex, user testing demonstrated that the experience can be easy, even for non-technical users. Just as with any software, achieving this kind of outcome depends on good user experience design, front-end implementation, and iterative feedback.

Blockchain as a component is always combined with other components, for key management, user interfaces, communication, and systems integration. Although architectural practice remains broadly similar, there are specific considerations when using blockchain. Architects must accommodate non-functional limitations on performance and security, and understand the constraints of data storage and smart contracts on immutable ledgers.

A number of research questions remain open:

- What is required to gain user trust in programmable money? How can non-technical users be equipped with a sufficient level of understanding of the policies attached to the money? These questions may pose the biggest risk for adoption: it can reasonably be assumed that a lack of understanding would result in lower willingness to adopt and accept programmable money, which in turn would hamper network effects of adoption and possibly lower valuation by parts of an economy.
- How can horizontal scaling be achieved for nodes creating transactions for a single blockchain account (as per the challenge mentioned in Sect. 5.2)?
- Is there a broader opportunity for new solutions that use dynamic rule-based systems for digital assets or other high-integrity constructs?
- What new concepts of money might be enabled, such as "values-based money", and what are their economic properties?
- Can the concept of programmable money be applied to implement alternative money systems, which may have specific desirable economic properties?

As for further uses of this technology, we foresee the following potential areas of application. First, as detailed in Royal et al. (2018, Chapter 8), *public policy programs* other than NDIS could be supported. Programmable money could support funding from public entities which is person-centred (like NDIS), cross-jurisdictional (e.g., healthcare treatment across state and territory boundaries), or outcomes-based (where service providers get paid based on outcome, not effort or cost). Second, programmable money could also be made *self-taxing*, such that taxes applied to a transaction automatically consider rebates, taxes, levies, etc. for the specific current situation in which the transaction takes place. Third, programmable money could also be used by individuals to codify their own *personal or financial goals*, and help them achieve those goals, e.g., through a personal (smart) savings plan or diet plan. Fourth, *additional business and not-for-profit use cases* can be supported, such as insurance payouts restricted to replacement or repair costs, simplified spending delegations, and donation management in not-for-profit organizations. Fifth, *values-based money* could allow directing the flow of money to further societal values, as discussed at various points throughout the paper. Finally, programmable

money could be used to realize *dynamic valuation of money*, based on age of the money, geography of the transacting parties, or other aspects. Among others, *demurrage money* which, as proposed by Gesell (1916), gradually loses value if it is not transacted on occasion could be realized with programmable money. Alternatively, to stimulate spending in a crisis situation, the local government could issue money that can only be spent locally and only for a limited period of time.

Acknowledgements We thank the co-authors of the project report, the project team, reference group, and all who helped us understand the NDIS and test the system. We further thank Michael Burda as well as the reviewers, editors, and discussants who contributed to improvements of this manuscript.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

Avital, M., Hedman, J., & Albinsson, L. (2017). Smart money: Blockchain-based customizable payments system. *Dagstuhl Reports, 7*(3), 104–106.

BIS. (2020). Central bank digital currencies: foundational principles and core features. Report, Bank of International Settlements

BoE. (2020). Central bank digital currency: Opportunities, challenges and design. Discussion paper, Bank of England

Bratanova, A., Devaraj, D., Horton, J., Naughtin, C., Kloester, B., Trinh, K., Weber, I., & Dawson, D. (2019). Blockchain 2030: A look at the future of blockchain in Australia. Tech. rep., Data61, CSIRO, Brisbane, Australia

Burda, M. C. (2021). Valuing cryptocurrencies: Three easy pieces. Working paper, Humboldt University, Berlin, Germany, April. https://www.wiwi.hu-berlin.de/de/forschung/irtg/results/discussion-papers/discussion-papers-2017-1/irtg1792dp2021-011.pdf. Accessed 14 Jul 2021

Elsden, C., Feltwell, T., Lawson, S., & Vines, J. (2019a). Recipes for programmable money. In: Proceedings of CHI. ACM. https://dl.acm.org/doi/10.1145/3290605.3300481.

Elsden, C., Trotter, L., Harding, M., Davies, N., Speed, C., Vines, J. (2019b). Programmable donations: Exploring escrow-based conditional giving. In: Proceedings of CHI. ACM. https://dl.acm.org/doi/10.1145/3290605.3300609.

Gesell, S. (1916). Die Natürliche Wirtschaftsordnung. In Walker, K. (ed.) (1949), 9th ed. Zitzmann.

Harvey, C. R. (2014). Bitcoin myths and facts. Available at SSRN 2479670

Harvey, C. R. (2016). Cryptofinance. Available at SSRN 2438299

Harvey, C. R., Ramachandran, A., Santoro, J. (2020). Defi and the future of finance. Available at SSRN 3711777. https://www.wiley.com/en-ca/DeFi+and+the+Future+of+Finance-p-9781119836025

Hedman, J. (2019). Smart money. Stockholm: Presentation at the Mobey Forum.

Jevons, W. (1875). *Money and the mechanism of exchange*. C. Kegan Paul.

Kolehmainen, T., Laatikainen, G., Kultanen, J., Kazan, E., Abrahamsson, P. (2021). Using blockchain in digitalizing enterprise legacy systems: An experience report. In: Klotins, E., Wnuk, K. (eds.) Software Business, pp. 70–85.

Lin, M. B., Khowaja, K., Chen, C., & Härdle, W. K. (2021). Blockchain mechanism and distributional characteristics of cryptos. In: Book Series: Advances in Quantitative Analysis of Finance & Accounting (AQAFA), *18* Forthcoming.

Magazzeni, D., McBurney, P., & Nash, W. (2017). Validation and verification of smart contracts: A research agenda. *Computer, 50*(9), 50–57.

Majuri, Y. (2019). Overcoming economic stagnation in low-income communities with programmable money. *The Journal of Risk Finance, 20*(5), 594–610.

Mendling, J., Weber, I., Aalst, W. v d, et al. (2018). Blockchains for business process management challenges and opportunities. *ACM Transactions on Management Information Systems (TMIS), 9*(1), 4:1-4:16. https://doi.org/10.1145/3183367

Offenberg, J. P. (2007). Markets: gift cards. *Journal of Economic Perspectives, 21*(2), 227–238.

Petukhina, A., Trimborn, S., Härdle, W. K., & Elendner, H. (2021). Investing with cryptocurrencies evaluating their potential for portfolio allocation strategies. *Quantitative Finance, 0*(0), 1–29.

Rodrigues, J., & Cardoso, A. (2019). Blockchain in smart cities: An inclusive tool for persons with disabilities. In: Smart City Symposium Prague (SCSP'19), pp. 1–6 https://doi.org/10.1109/SCSP.2019.8805708.

Royal, D., Rimba, P., Staples, M., Gilder, S., Tran, A., Williams, E., Ponomarev, A., Weber, I., Connor, C., & Lim, N. (2018). Making money smart: Empowering NDIS participants with blockchain technologies. Report by CSIRO and Commonwealth Bank of Australia, https://data61.csiro.au/en/Our-Research/Our-Work/SmartMoney. Accessed 18 Feb 2021.

Schilling, L., & Uhlig, H. (2019a). Some simple bitcoin economics. *Journal of Monetary Economics, 106,* 16–26.

Schilling, L. M., & Uhlig, H. (2019b). Currency substitution under transaction costs. *AEA Papers and Proceedings, 109,* 83–87.

Weber, I., & Staples, M. (2021). Programmable money: Next-generation conditional payments using blockchain—keynote paper. In: CLOSER'21: International Conference on Cloud Computing and Services Science (pp. 7–14). https://www.proceedings.com/content/061/061616webtoc.pdf. ISBN: 978-1-7138-3981-1

Weber, I., Xu, X., Riveret, R., Governatori, G., Ponomarev, A., & Mendling, J. (2016). Untrusted business process monitoring and execution using blockchain. In: Proceedings of BPM. Springer. https://link.springer.com/chapter/10.1007/978-3-319-45348-4_19.

Xu, X., Weber, I., & Staples, M. (2019). *Architecture for blockchain applications*. Springer. https://doi.org/10.1007/978-3-030-03035-3