Programmable Money: Next-generation Conditional Payments using Blockchain

Ingo Weber[1][a] and Mark Staples[2][b]
[1]Chair of Software and Business Engineering, Technische Universitaet Berlin, Germany
[2]CSIRO's Data61, Australia and School of Computer Science & Engineering, UNSW, Australia

Keywords: Blockchain, DLT, Smart Money, Programmable Money, Conditional Payment, Keynote.

Abstract: Conditional payments allow the transfer of money only when predefined 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 under which conditions money may be spent, and can be automatically checked by the money as it is spent. Policies can be dynamically added to and removed from money as it flows through an economy. Here we report on some of our experiences and insights regarding blockchain architecture, software engineering with blockchain, and blockchain-based programmable money. Selected open research questions are listed at the end of the paper.

1 INTRODUCTION

Disruptive technologies break assumptions about limitations of systems and business models, creating new 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 decentralised shared ledgers. Software engineering for blockchain-based systems must adapt to respective new architectural and development challenges (Xu et al., 2019).

Bitcoin demonstrated a solution to the problem of digital cash, using a single *logically-centralised ledger* of cryptocurrency transactions operated in an *organisationally-decentralised and physically-distributed* way by a collective of thousands of nodes. 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.

In collaboration with a large bank, we conducted the "Smart Money" project, to conceptualize, implement, and evaluate systems for blockchain-based programmable money (Royal et al., 2018). This is a novel concept where policies can be dynamically associated with money, and checked and updated as that money is transferred. The project was motivated by 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

[a] https://orcid.org/0000-0002-4833-5921
[b] https://orcid.org/0000-0003-3284-5385
[1]In this article, we say 'blockchain' as a short-hand for blockchain and other distributed ledger technologies.

Weber, I. and Staples, M.
Programmable Money: Next-generation Conditional Payments using Blockchain.
DOI: 10.5220/0010535800070014
In *Proceedings of the 11th International Conference on Cloud Computing and Services Science (CLOSER 2021)*, pages 7-14
ISBN: 978-989-758-510-4

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 et 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 tradeoffs (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 et 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 Section 2. Then we discuss implementation concerns in Section 3 and summarise the evaluation in Section 4. Finally, lessons learnt are described in Section 5 before the paper is concluded in Section 6.

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 removed from money that flows through a payments system. 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 new policies can be attached to money by its owner, policies by default remain attached to money as it is paid, and policies can update or remove themselves 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 realise the desired features, blockchain technology with smart contract capabilities forms a natural base. Alternative, centralized realisations without blockchain as a base can be considered, and have been in the project's analytical evaluation (Royal et 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.,

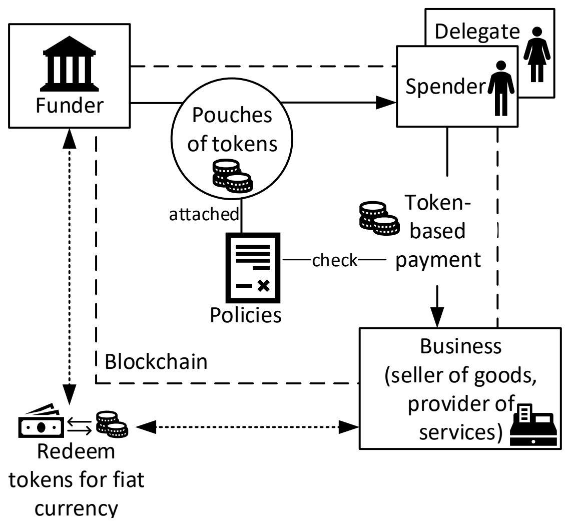


Figure 1: Solution overview, showing policies attached to money (pouches of tokens), transferable between parties, and optionally redeemable.

one government agency in charge) for the multiple use cases, (ii) end users would not need 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 Section 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 realisation of multiple use cases.

An overview of the chosen on-chain architecture is depicted in Figure 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 payment only succeeds 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 Section 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. 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 them-

```
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.

selves 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.

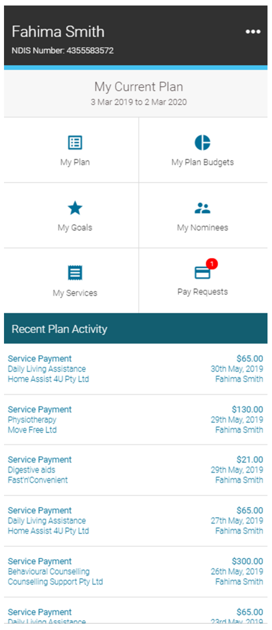
Programmable money enables powerful forms of conditional payments. Dynamically-modifiable policies provide a high degree of flexibility. Conditions can be about the service or asset paid for, a previous payment, or the status of another account or money (thereby also comprising a form of *higher-order 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.

3 IMPLEMENTATION

We implemented the concept in a working prototype. The research organisation (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 et al., 2018). Selected screenshots of the prototype’s UI are shown in Figure 2. Specifically, the overview screen for NDIS participants (Figure 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 (Figure 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 (Figure 2c), a map and a list view show available providers, including their star rating. Payment requests (Figure 2d) list the details of the request and allow the user to authorise 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 summarise the back-end implementation and deployment. 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. 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 (parametrised) 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 del-


Fahima Smith
NDIS Number: 4355583572

My Current Plan
3 Mar 2019 to 2 Mar 2020

Recent Plan Activity

Service Payment — $65.00
Daily Living Assistance — 30th May, 2019
Home Assist 4U Pty Ltd — Fahima Smith

Service Payment — $130.00
Physiotherapy — 29th May, 2019
Move Free Ltd — Fahima Smith

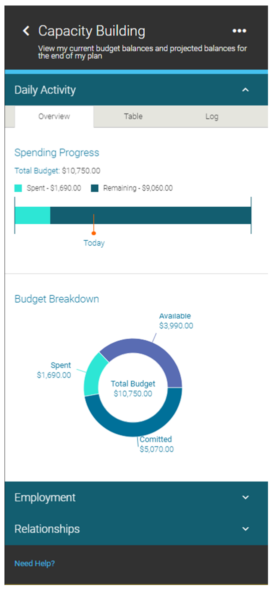
Service Payment — $21.00
Digestive aids — 29th May, 2019
Fast'n'Convenient — Fahima Smith

Service Payment — $65.00
Daily Living Assistance — 27th May, 2019
Home Assist 4U Pty Ltd — Fahima Smith

Service Payment — $300.00
Behavioural Counselling — 26th May, 2019
Counselling Support Pty Ltd — Fahima Smith

Service Payment — $65.00
Daily Living Assistance — 23rd May, 2019

(a) Overview screen

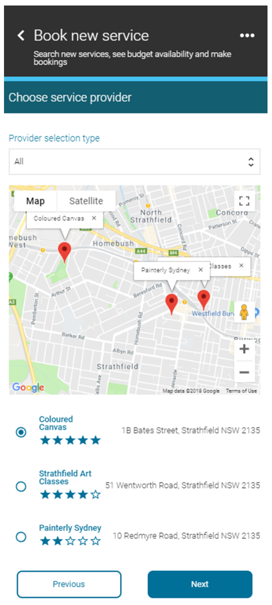

Capacity Building
View my current budget balances and projected balances for the end of my plan

Daily Activity

Spending Progress
Total Budget: $10,750.00
🟦 Spent - $1,690.00 ■ Remaining - $9,060.00

Budget Breakdown

(b) Budget overview

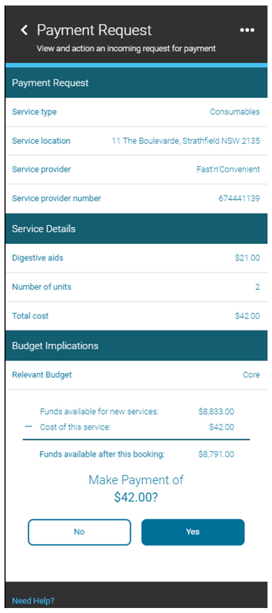

Book new service
Search new services, see budget availability and make bookings

Choose service provider

Provider selection type: All

[✓ Coloured Canvas] ★★★★★ 1B Bates Street, Strathfield NSW 2135
Strathfield Art Classes ★★★★☆ 51 Wentworth Road, Strathfield NSW 2135
Painterly Sydney ★★☆☆☆ 10 Redmyre Road, Strathfield NSW 2135

(c) Booking a service


Payment Request
View and action an incoming request for payment

Payment Request	
Service type	Consumables
Service location	11 The Boulevarde, Strathfield NSW 2135
Service provider	Fast'n'Convenient
Service provider number	674441139
Service Details	
Digestive aids	$21.00
Number of units	2
Total cost	$42.00
Budget Implications	
Relevant Budget	Core

Funds available for new services: $8,833.00
— Cost of this service: $42.00
Funds available after this booking: $8,791.00

Make Payment of
$42.00?

(d) Payment request

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

egates in the project interacted with the blockchain through a mobile app, which also managed the key-pairs for authorisation. Integration with the frontend 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 centralised NDIS system used in production, payment success rates are reported publicly[2] 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.

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 2), 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

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

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

We report on learnings about programmable money on blockchain, and about software engineering for blockchain-based systems.

5.1 Programmable Money on Blockchain

The project report (Royal et 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 conventional technology. Although uncommon today, centralised 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.

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.

Finally, we note that 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.

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 compute platform, and often as a mechanism for asset control and management (Xu et al., 2019, Chapter 5). Software engineering with blockchain is similar in many ways to conventional technologies, but there are differences.

Some things remain similar for software engineering with blockchain:

- 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 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 things 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 associated 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. 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 authorisation in blockchain systems, and this is more complex and risk-prone architecturally and for users than centralised 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 AND OPEN RESEARCH QUESTIONS

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 Section 2. We have described some of our experiences and learnings from our "Smart Money" project (Royal et 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?
- How can horizontal scaling be achieved for nodes

creating transactions for a single blockchain account (as per the challenge mentioned in Section 5.2)?

- Is there a broader opportunity for new solutions that use dynamic rule-based systems for digital assets or other high-integrity constructs?
- Can the concept of programmable money be applied to implement alternative money systems, which may have specific desirable economic properties?

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.

REFERENCES

Avital, M., Hedman, J., and 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.

Elsden, C., Feltwell, T., Lawson, S., and Vines, J. (2019a). Recipes for programmable money. In *Proceedings of CHI*. ACM.

Elsden, C., Trotter, L., Harding, M., Davies, N., Speed, C., and Vines, J. (2019b). Programmable donations: Exploring escrow-based conditional giving. In *Proceedings of CHI*. ACM.

Hedman, J. (2019). Smart money. Presentation at the mobey forum, Stockholm.

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

Magazzeni, D., McBurney, P., and 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 et al., J. (2018). Blockchains for business process management – challenges and opportunities. *ACM Transactions on Management Information Systems (TMIS)*, 9(1):4:1–4:16.

Rodrigues, J. and Cardoso, A. (2019). Blockchain in smart cities: An inclusive tool for persons with disabilities. In *Smart City Symposium Prague (SCSP'19)*, pages 1–6.

Royal, D., Rimba, P., Staples, M., Gilder, S., Tran, A., Williams, E., Ponomarev, A., Weber, I., Connor, C., and 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: 18/2/2021.

Weber, I., Xu, X., Riveret, R., Governatori, G., Ponomarev, A., and Mendling, J. (2016). Untrusted business process monitoring and execution using blockchain. In *Proceedings of BPM*. Springer.

Xu, X., Weber, I., and Staples, M. (2019). *Architecture for Blockchain Applications*. Springer.