Um Estudo Inicial sobre a Importância de Simular Contratos Inteligentes em Blockchain

Alan Nascimento Gomes[1], **Emanuel Ferreira Coutinho**[1]

[1]Programa de Pós-Graduação em Computação (PCOMP)
Universidade Federal do Ceará (UFC) – Quixadá – CE – Brasil

alanng@alu.ufc.br, emanuel.coutinho@ufc.br

Abstract. *With the growing use of computer systems in different sectors of society, new technologies have emerged, and with them many challenges. Blockchain is a peer-to-peer network that stores a chain of blocks, comparable to a ledger, where each interested party has an identical, transparent, and immutable copy. Smart contracts enable dynamic agreements on the blockchain with greater confidence in the exchange of digital assets. However, despite the use of smart contracts bringing benefits, there is a need to be careful with threats, such as vulnerabilities and errors in the business logic. The purpose of this research is to discuss the importance of simulating smart contracts in blockchain. For this, some research questions were designed, data were obtained from a literature review and some categories of aspects related to blockchain simulation were identified.*

Resumo. *Com a crescente utilização de sistemas computacionais em diversos setores da sociedade, novas tecnologias surgiram, e com elas vários desafios. Blockchain é uma rede ponto a ponto que armazena uma cadeia de blocos, comparável a um livro público, onde cada parte interessada possui uma cópia idêntica, transparente e imutável. Os contratos inteligentes possibilitam na blockchain acordos dinâmicos e com maior confiança na troca de ativos digitais. Porém, apesar da utilização de contratos inteligentes trazer benefícios, há a necessidade de cuidados em relação a ameaças, como vulnerabilidades e erros na lógica de negócio. O objetivo desta pesquisa é discutir sobre a importância de simular contratos inteligentes em blockchain. Para isso, algumas questões de pesquisa foram projetadas, dados foram obtidos de uma revisão da literatura e algumas categorias de aspectos relacionados à simulação de blockchain foram identificados.*

1. Introdução

Com a crescente utilização de sistemas computacionais em diversos setores da sociedade, requisitos básicos para o seu bom funcionamento são necessários. Dentre essas características pode-se citar algumas, como: segurança, privacidade, escalabilidade e sustentabilidade [Xu et al. 2016]. Tendo em vista essas condições, a tecnologia blockchain surgiu com um grande potencial para ser aplicada em diferentes tipos de sistemas e campos. Um dos principais desafios dos contratos inteligentes é garantir sua exatidão e segurança [Akca et al. 2019], e por isso, em um cenário com utilização dessa tecnologia é indispensável a realização de testes para assegurar que a aplicação esteja se comportando conforme o esperado.

Blockchain é uma rede *peer-to-peer* que armazena uma cadeia de blocos [Bhaskar e Chuen 2015]. Os dados armazenados na rede são mantidos por vários nós que contém um *log* completo de todas as transações. O armazenamento dos registros é comparável a um livro público, onde cada parte interessada possui uma cópia idêntica, transparente e imutável da cadeia [Thakkar et al. 2018]. Estas características estão presentes devido a propriedades pertencentes a cada bloco, por exemplo: a presença de um ponteiro para o bloco anterior, assinatura de data e hora e um atributo denominado *nonce*, que garante a integridade dos dados.

Blockchain chegou a uma nova fase com a implantação de contratos inteligentes. Estes possuem o objetivo de possibilitar a utilização de redes blockchain para acordos dinâmicos e com maior confiança na troca de ativos digitais [Swan 2015]. Os contratos inteligentes funcionam como códigos executáveis que agem conforme as condições do acordo entre as partes interessadas, possibilitando a inexistência de uma entidade intermediária para garantia das transações. Apesar da utilização de contratos inteligentes trazer benefícios para as soluções, existe a necessidade de cuidados em relação a possíveis ameaças ligadas ao uso de contratos inteligentes. Singh et al. (2020) levantaram algumas questões e vulnerabilidades enfrentados por contratos inteligentes. A verificação da funcionalidade foi o aspecto de vulnerabilidade mais comum encontrados em contratos inteligentes. Privacidade, segurança, escalabilidade e confiabilidade são possíveis pontos sujeitos a produzir ações inesperadas durante a execução de contratos inteligentes.

Além disso, um contrato inteligente que foi inserido na blockchain sintaticamente correto, mas com problemas nas regras de negócio, poderá ser executado normalmente, provocando problemas para a aplicação e inserindo dados incoerentes com as regras de negócio. Esse código incorreto não pode ser apagado, o que leva a uma discussão para testes em contratos inteligentes e a simulação para identificação de problemas.

O objetivo desta pesquisa é discutir a importância de simular contratos inteligentes em blockchain. Para isso, as seguintes questões de pesquisa (QP) foram projetadas:

- QP1 - Por que simular um contrato inteligente?
- QP2 - O que simular em um contrato inteligente?
- QP3 - Como simular um contrato inteligente?

Para responder a essas questões, procurou-se na literatura trabalhos que tratassem de aspectos relacionados a blockchain e simulação. Em seguida, alguns trabalhos foram descritos, e algumas categorias foram propostas para poder facilitar a resposta das questões de pesquisa. Como contribuições tem-se: (i) uma discussão sobre aspectos de contratos inteligentes que devem ser considerados para uma maior confiança e desempenho; e (iii) identificação de ferramentas para apoiar a simulação de contratos inteligentes.

O restante do trabalho está dividido nas seguintes seções: a Seção 2 apresenta uma fundamentação teórica sobre blockchain; a Seção 3 apresenta algumas iniciativas em simulação de contratos inteligentes; a Seção 4 retorna às questões de pesquisa e discute os resultados; e por fim, na Seção 5 as considerações finais são apresentadas.

2. Fundamentação Teórica

2.1. *Blockchain*

Blockchain é uma sequência de blocos que contém um registro completo de transações como um livro público, indicando a ordem na qual as transações ocorre-

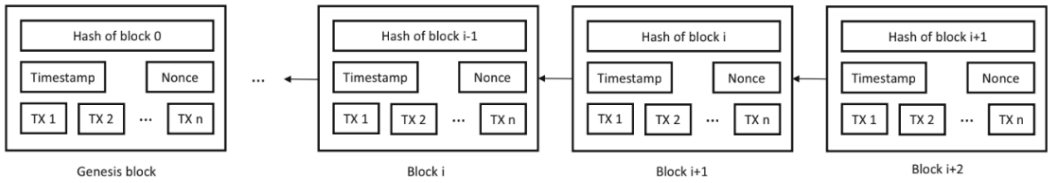


Figura 1. *Blockchain* e seus blocos [Nofer et al. 2017][Zheng et al. 2017]

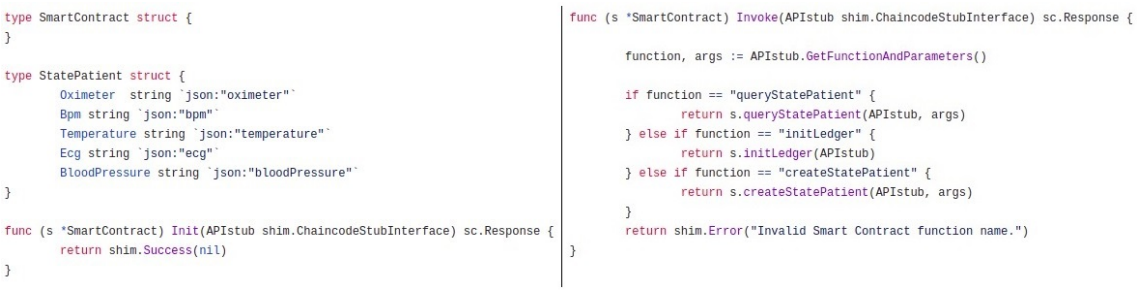
ram [Bhaskar e Chuen 2015]. A Figura 1 exibe uma *blockchain* com um bloco recém validado apontando para o bloco imediatamente anterior gerado. Cada bloco da cadeia confirma a integridade do anterior, e também todo o caminho de volta até o primeiro bloco (bloco de gênese). Uma *blockchain* consiste em um conjunto de dados compostos por uma cadeia de pacotes de dados (blocos) onde um bloco compreende transações múltiplas [Nofer et al. 2017]. Ela é estendida por cada bloco adicional e, portanto, representa um registro geral completo do histórico de transações. Estes blocos podem ser validados pela rede usando mecanismos criptográficos. Além das transações, cada bloco contém um carimbo de data e hora (*timestamp*), o valor de *hash* do bloco anterior (pai), e um "*nonce*", que é um número aleatório para verificar o *hash*. Este conceito garante a integridade de toda a *blockchain* até o primeiro bloco.

Os valores do *hash* são únicos e fraudes podem ser efetivamente prevenidas, uma vez que as mudanças em um bloco na cadeia mudariam imediatamente o respectivo valor do *hash* [Nofer et al. 2017]. Caso a maioria dos nós da rede concordarem por meio de um mecanismo de consenso sobre a validade das transações em um bloco e sobre a validade do próprio bloco, então este bloco pode ser adicionado à cadeia. Portanto, novas transações não são automaticamente adicionadas ao registro. Em vez disso, o processo de consenso garante que essas transações sejam armazenadas em um bloco por certo tempo (por exemplo 10 minutos na *blockchain Bitcoin*) antes de serem transferidas para o livro-razão. Após este processo, as informações na *blockchain* não podem mais ser alteradas. No caso do *Bitcoin*, os blocos são criados pelos chamados "mineradores", que são recompensados com *Bitcoins* pela validação dos blocos.

2.2. Contratos Inteligentes

O conceito de contrato inteligente foi introduzido por Nick Szabo em 1994, definido como um protocolo de transação computadorizado que executa os termos de um contrato. Szabo sugeriu traduzir cláusulas contratuais, como garantias e títulos, em código e incorporá-las em propriedades (hardware ou software) que possam se autoaplicar, de modo a minimizar a necessidade de intermediários confiáveis entre as partes envolvidas na transação, e a ocorrência de exceções maliciosas ou acidentais [Szabo 1994].

Dentro do contexto da *blockchain*, os contratos inteligentes são um fluxo de valor baseado em certos termos e condições, como contratos no mundo real. A única diferença é que eles são completamente digitais, significando que um pequeno código é armazenado na *blockchain*. Existem diferentes plataformas de *blockchain* que podem ser utilizadas para desenvolver contratos inteligentes, sendo a *Ethereum* a mais utilizada [Alharby e van Moorsel 2017]. Os contratos inteligentes funcionam como *scripts* armazenados. Como residem na cadeia, eles possuem um endereço exclusivo. Pode-se acionar um contrato inteligente endereçando uma transação para ele, onde em seguida, ele executa



```
type SmartContract struct {
}

type StatePatient struct {
        Oximeter  string `json:"oximeter"`
        Bpm string `json:"bpm"`
        Temperature string `json:"temperature"`
        Ecg string `json:"ecg"`
        BloodPressure string `json:"bloodPressure"`
}

func (s *SmartContract) Init(APIstub shim.ChaincodeStubInterface) sc.Response {
        return shim.Success(nil)
}
```

```
func (s *SmartContract) Invoke(APIstub shim.ChaincodeStubInterface) sc.Response {

        function, args := APIstub.GetFunctionAndParameters()

        if function == "queryStatePatient" {
                return s.queryStatePatient(APIstub, args)
        } else if function == "initLedger" {
                return s.initLedger(APIstub)
        } else if function == "createStatePatient" {
                return s.createStatePatient(APIstub, args)
        }
        return shim.Error("Invalid Smart Contract function name.")
}
```

Figura 2. Trecho de um contrato inteligente em GO para o Hyperledger Fabric

de forma independente da forma que foi escrito, em qualquer nó da rede, de acordo com os dados que foram incluídos no acionamento da transação [Christidis e Devetsikiotis 2016].

Contratos inteligentes são criados sobre uma plataforma de criptomoeda (e.g. *Ethereum*). Uma criptomoeda é um sistema descentralizado para interagir com dinheiro virtual em um livro compartilhado de forma global. Os usuários transferem dinheiro e interagem com contratos por meio da publicação de dados assinados, denominados de transações da rede de criptomoedas. A rede consiste em nós chamados mineradores que propagam informações, armazenam dados, e atualizam os dados aplicando transações. A Figura 2 exibe um trecho exemplo de um contrato inteligente escrito em GO para a blockchain Hyperledger Fabric.

3. Iniciativas de Simulação de Contratos Inteligentes

Para ilustrar alguns exemplos ou iniciativas de simulação de contratos inteligentes em blockchain, alguns trabalhos da literatura foram descritos. Destaca-se que estes são apenas alguns trabalhos, sendo que não foi conduzido um profundo levantamento da literatura. A forma de seleção dos trabalhos ocorreu de maneira *ad hoc*. Esses trabalhos possibilitaram a identificação de categorias de aspectos a serem considerados para a simulação de contratos inteligentes em blockchain.

Abdellatif e Brousmiche (2018) propuseram uma abordagem de modelagem formal para verificar o comportamento de um contrato inteligente em seu ambiente de execução. O formalismo foi aplicado em um exemplo concreto de contrato inteligente e suas violações foram analisadas com uma abordagem de verificação de modelo estatístico. Utilizou-se o framework BIP (*Behavior Interaction Priorities*). Ao aplicar esse formalismo, modelou-se o comportamento e interações com o ambiente de execução. A simulação desses comportamentos no BIP e a análise de seus resultados usando a verificação de modelo estatístico permitiram revelar cenários onde o comportamento do contrato inteligente pode ser violado por *hackers*.

Destefanis et al. (2018) defenderam a necessidade de uma disciplina de Engenharia de Software para blockchain, abordando os problemas colocados pela programação de contratos inteligentes e outros aplicativos executados em blockchain. Um estudo de caso onde um defeito descoberto em uma biblioteca de contratos inteligentes foi analisado, verificando que talvez uma programação “insegura” tenha permitido um ataque ao Parity, um aplicativo de carteira digital. O código-fonte do Parity e da biblioteca foram analisados e discutiu-se como melhores práticas reconhecidas podem mitigar esse mau comportamento prejudicial de software, se adotadas e adaptadas. Também refletiu-se so-

bre a especificidade do desenvolvimento de contratos inteligentes, o que torna algumas abordagens existentes insuficientes, exigindo a definição de uma Engenharia de Software específica para blockchain.

Liu et al. (2018) apresentaram o ReGuard, um analisador baseado em lógica fuzzy para detectar automaticamente defeitos de reentrada em contratos inteligentes na Ethereum, gerando código em c++. O ReGuard realiza testes em contratos inteligentes gerando iterativamente transações aleatórias. Com base nos rastreamentos de tempo de execução, o ReGuard identifica dinamicamente as vulnerabilidades de reentrada. Cinco contratos inteligentes da Ethereum foram avaliados.

Parizi et al. (2018) argumentaram que para realizar mais prontamente a aplicação de contratos inteligentes com segurança e privacidade, primeiro deve-se entender suas vulnerabilidades antes da implementação generalizada. Uma avaliação empírica de ferramentas de análise de segurança automática de código aberto para a detecção de vulnerabilidades de segurança de contratos inteligentes foi conduzida, sobre a Ethereum e escritos em Solidity (Oyente, Mythril, Securify e SmartCheck). As ferramentas foram testadas em dez contratos inteligentes, tanto na eficácia da vulnerabilidade quanto na precisão dos pontos de vista de detecção verdadeiros. Os resultados mostraram que a ferramenta SmartCheck é estatisticamente mais eficaz do que as outras ferramentas automatizadas de teste de segurança com nível de significância de 95%. Quanto à precisão, o Mythril foi significativamente preciso com a emissão do menor número de falsos alarmes entre as ferramentas. Esses resultados podem implicar que o SmartCheck pode ser atualmente a ferramenta de teste de segurança estática mais eficaz para contratos inteligentes Solidity na blockchain Ethereum, porém talvez menos precisa que o Mythril.

Akca et al. (2019) analisaram um desafio dos contratos inteligentes, que é garantir sua exatidão e segurança. Para enfrentar esse desafio, uma técnica automatizada (SolAnalyser) foi desenvolvida, para detecção de vulnerabilidades em contratos inteligentes Solidity que usa dados estáticos e analise dinâmica. A eficácia da técnica em revelar as vulnerabilidades foi avaliada e comparou-se com cinco ferramentas de análise populares existentes: Oyente, Securify, Maian, SmartCheck e Mythril.

Chapman et al. (2019) apresentaram o Deviant, uma ferramenta de teste de mutação para contratos inteligentes Solidity, que gera automaticamente mutantes de um determinado projeto Solidity e executa todos os mutantes nos testes fornecidos para avaliar sua eficácia. Para simular várias falhas nos contratos inteligentes do Solidity, o Deviant fornece operadores de mutação para todos os recursos exclusivos do Solidity de acordo com o modelo de falha do Solidity, além das construções de programação tradicionais. O Deviant foi utilizado para avaliar a eficácia dos testes em três projetos do Solidity. Os resultados indicaram que esses testes ainda não alcançaram altas pontuações de mutação e que um conjunto de testes adequado para os critérios de declaração e cobertura de ramificação dos contratos inteligentes Solidity não fornece necessariamente uma garantia de alto nível de qualidade de código. Tais observações oferecem diretrizes importantes para os desenvolvedores do Solidity implementarem testes mais eficazes para entregar um código confiável e reduzir o risco de perdas financeiras.

Abreu et al. (2020) apresentaram uma ferramenta para aplicação no contexto educacional de registro de diplomas, baseado na blockchain Ethereum. Não foi utilizada a

rede real de produção da Ethereum, e sim uma rede de testes, a Ropsten. Também foi utilizado o Infura, que é um site para interagir com contratos inteligentes porque precisamos estar conectados a Ethereum, e o Etherscan, que é uma ferramenta para explorar um blockchain, permitindo a análise transações na Ethereum para auxiliar na validação durante o desenvolvimento de aplicativos. Por meio da IDE Remix, utilizando códigos para contratos inteligentes escritos em Solidity, foi possível testar e simular situações da execução das funções da aplicação, antes de pô-las em produção.

Coutinho et al. (2020) simularam os custos financeiros do uso da blockchain Ethereum com o Remix. A ideia foi simular os custos de *gas* e *ether* resultantes de transações, e assim permitir analisar o impacto da quantidade e tamanho das transações no custo. Os custos de *gas* e *ether* podem variar conforme o tamanho do código do método do contrato inteligente e da quantidade de chamadas. O *gas* é a unidade de medida do poder computacional na Ethereum, e o *ether* é para a medição e pagamento pelo custo computacional no Ethereum. A aplicação apresentada no trabalho foi relacionada à doações de valores. Esses valores podem ser convertidos para uma moeda financeira, como real ou dólar, e assim analisar os custos financeiros de uma aplicação blockchain.

Liu et al. (2020) discutiram que as ferramentas atuais de teste e análise carecem de suporte para esses contratos, que demonstram comportamentos de estado e exigem tratamento especial na garantia de qualidade. Em seu trabalho, os autores apresentaram uma plataforma de teste baseada em modelo, chamada ModCon, com modelos especificados pelo usuário para definir oráculos de teste, orientar a geração de teste e medir a adequação do teste. ModCon é baseado na web e suporta plataformas blockchain sem permissão e com permissão. O uso e os principais recursos do ModCon foram aplciados em aplicativos de contratos inteligentes corporativos reais.

Nascimento Gomes e Coutinho (2021) utilizaram o Hyperledger Fabric para a prototipação de uma aplicação de registros médicos, simulando uma aplicação de prontuário para exibir situações de risco de pacientes. Para testar a aplicação, cargas de trabalho geradas manualmente e automatizadas foram utilizadas. Assim, pontos de melhoria e de potenciais problemas na aplicação e infraestrutura puderam ser identificados e melhores projetados. É importante rastrear se os dados estão sendo inseridos corretamente, e isso é uma consequência de um contrato inteligente correto tanto sintaticamente quanto semanticamente (regras de negócio corretamente implementadas).

Abreu et al. (2021) conduziram uma avaliação de desempenho na blockchain Ethereum por meio de um aplicativo desenvolvido para analisar o custo e o tempo das transações que armazenam caracteres texto em uma blockchain. Para atender ao objetivo proposto, uma aplicação para realização de transações com inclusão de dados e consulta em um blockchain foi projetada, coletando dados de tempo e custo. As principais conclusões foram: a plataforma Ethereum se mostrou inconstante em relação ao tempo de processamento das transações na blockchain e a aplicação desenvolvida com base na blockchain pode fornecer um mecanismo para avaliar operações do tipo texto na rede Ethereum. Esses resultados reforçam a importância de simular contratos inteligentes nos aspectos tempo e custo.

Tabela 1. Comparativo entre trabalhos: Trabalho, Verificação formal (VF), Vulnerabilidades (VUL), Testes (TES), Implantação (IMP), Desempenho (DES), Aplicação (APL) e Ferramentas

TRABALHO	VF	VUL	TES	IMP	DES	APL	FERRAMENTAS
[Abdellatif e Brousmiche 2018]	x	x					BIP
[Akca et al. 2019]		x	x				Ethereum, Solidity, SolAnalyser, Oyente, Securify, Maian, SmartCheck and Mythril
[Destefanis et al. 2018]		x	x				Parity, Ethereum
[Liu et al. 2018]		x	x				c++, Reguard, Etherscan, Oyente
[Parizi et al. 2018]		x	x				Remix, Ethereum, Solidity, Oyente, Mythril, Securify, SmartCheck
[Chapman et al. 2019]			x				Solidity, Ethereum, Deviant, Truffle
[Abreu et al. 2020]			x	x		x	Remix, Ethereum, Solidity, Etherscan, Ropsten, Infura
[Coutinho et al. 2020]					x	x	Remix, Ethereum, Solidity
[Liu et al. 2020]			x				Javascript, ModCon
[Nascimento Gomes e Coutinho 2021]						x	Hyperledger Fabric, GO,
[Abreu et al. 2022]					x		Ethereum, Solidity, Remix,

4. Categorias e Questões de Pesquisa

A ideia desta pesquisa é ter uma discussão sobre a importância de simular contratos inteligentes em blockchain. Por simular um contrato inteligente também pode-se considerar teste e modelagem, e consequentemente avaliar protótipos resultantes antes de pô-los em produção ou testes mais robustos. E para a simulação em blockchain, é necessário conhecer e explorar as características da infraestrutura, das linguagens de programação e das diversas plataformas e ferramentas relacionadas, entendendo seus benefícios, problemas e consequências. Como simulação abrange diversos aspectos, é necessário focar no que seria adequado para a realidade da blockchain.

A Tabela 1 apresenta algumas categorias que foram identificadas, baseadas na leitura dos artigos, e com o intuito de sinalizar áreas de atuação para simulação de contratos inteligentes em blockchain. As categorias foram: verificação formal, vulnerabilidades, teste, implantação, desempenho, aplicação e ferramentas.

Apenas um trabalho apontou verificação formal para contratos inteligentes. Esta área aparenta ser pouco explorada. Vulnerabilidades foi uma das categorias mais mencionadas. Além das possíveis vulnerabilidades da infraestrutura de rede da blockchain, problemas no código do contrato inteligente podem levar à grandes problemas para uma aplicação. Testes no contratos inteligentes foi bastante mencionado, e como nele residem regras de negócio, há uma demanda por testes específicos para blockchain. A implantação em si do contrato inteligente, se ele foi corretamente disponibilizado para a aplicação, se ele está funcional, o quanto ele pode ser mantido ou atualizado, são questões que devem ser consideradas para um ambiente de blockchain. Desempenho é muito relacionado à infraestrutura, como latência das operações e vazão das requisições, e impactam diretamente na qualidade das aplicações. Aplicação desde camadas físicas, de dados, de blockchain, de comunicação, e de interfaces de usuários devem ser bem integradas, e simulação nesses diversos níveis é algo a ser planejado. Ferramentas relacionadas ao desenvolvimento para blockchain existem várias. A forma como elas são integradas à ambientes mais tradicionais de desenvolvimento, com web e móvel, é algo que naturalmente vai ocorrer.

4.1. Por que simular um contrato inteligente?

Simulação traz diversos benefícios. Em relação a contratos inteligentes, considerando que é código fonte, naturalmente deve-se pensar em uma aplicação ou serviço de qualidade. E nesse contexto, simular contratos inteligentes colabora para ampliar a qualidade tanto da aplicação quanto do serviço. Ao se considerar que simular situações diversas onde um contrato inteligente pode ser utilizado, se conhece sobre o ambiente e infraestrutura, se conhece mais sobre o comportamento da aplicação. Assim, além de prevenir potenciais problemas, pode-se constantemente analisar o desempenho de todo o ecossistema de software que se beneficia das características da blockchain.

Também é necessário ter uma visão global da aplicação final ou regras de negócio. Como um contrato inteligente nada mais é do que código escrito em uma linguagem de programação, ele comporá uma aplicação. Isso envolve regras de negócio, que devem ser adequadas às necessidades do cliente. Nesse contexto também entram testes, que possuem reconhecida importância no desenvolvimento de sistemas.

Além disso, uma análise da viabilidade de soluções, tanto do ponto de vista técnico quanto financeiro colaboraria para a sustentabilidade da aplicação e evolução. Por fim, os riscos envolvidos também devem ser analisados.

4.2. O que simular em um contrato inteligente?

O que simular em um contrato inteligente vai muito em consonância ao que se pretende explorar. Pode ser sob uma perspectiva puramente técnica, como a sintaxe dos códigos, ou de negócio, por exemplo os requisitos funcionais plenamente atendidos, ou financeira, como o custo de manter uma aplicação blockchain em produção.

Uma vez um contrato inteligente implantado na blockchain, ele se torna imutável. Mesmo com erros, ele estará armazenado, não podendo ser alterado. Pode-se entretanto adicionar novos contratos inteligentes. Isso implica na importância de testes nos mais diversos níveis (unitários, integração, sistema, etc), por exemplo: testes de métodos do contrato inteligente, testes de inclusão e consulta na blockchain.

Em relação à infraestrutura da blockchain, há aspectos de redes de computadores, sistemas distribuídos e integração com a aplicação do usuário final ou com outros sistemas, sendo que tais aspectos devem ser testados ou simulados. Requisitos não funcionais também devem ser bem especificados, pois há um risco maior de problemas em sistemas blockchain do que em sistemas tradicionais, como apenas uma aplicação web ou móvel.

4.3. Como simular um contrato inteligente?

A questão do como simular é um desafio para blockchain. Existem diversas ferramentas para simular a infraestrutura, mas para contratos inteligentes não. O que se encontra são IDEs, como o Remix para a Ethereum, onde o usuário pode utilizar testes manuais ou automatizar, identificando erros no código e impedindo que problemas afetem a produção. Além disso, testes em produção implicariam no custo financeiro, caso a blockchain utilizada não seja gratuita. A identificação por ferramentas que auxiliem no processo de simulação é então essencial, além da definição do que simular de maneira mais sistemática. Como a blockchain possui essencialmente operações de inclusão e consulta, muito da simulação seria focando nessas operações, simulando as regras de negócio. E

a simulação de cargas de trabalho variadas pode colaborar com impactos em infraestrutura e custos financeiros, assim como testes de estresse e desempenho. Um exemplo é a utilização do Hyperledger Caliper para o projeto e execução de cargas de trabalho diversas utilizando as operações do contrato inteligente.

Diferentes linguagens de programação podem ser utilizadas para construir contratos inteligentes, como Solidity para a Ethereum e GO para o Hyperledger Fabric. Isso implica em testes nos mais variados níveis. Logo, ferramentas para testar aplicações, especialmente integrações e desempenho são necessárias para um ambiente de desenvolvimento em blockchain. No caso específico de vulnerabilidades, recomenda-se o uso de ferramentas de análise, como Oyente, Securify, Maian, SmartCheck e Mythril.

5. Considerações Finais

Esse trabalho é um ensaio inicial sobre pesquisas em simulação para blockchain, especificamente para contratos inteligentes. Nos contratos inteligentes é onde normalmente residem as regras de negócio da aplicação, sendo necessário uma atenção para evitar problemas em produção. Esta pesquisa trará benefícios para analistas de negócio que poderão planejar melhor seus contratos inteligentes, e analisar o desempenho e custos de uma aplicação que utilize recursos de blockchain. Os próximos passos da pesquisa consistem na identificação mais precisa do que seria simular blockchain. Isso envolve modelagem de sistemas, projeto de contratos inteligentes e ferramentas para suporte às atividades de simulação. Quais técnicas de simulação poderiam ser exploradas no contexto de contratos inteligentes também deve ser estudado. Além disso, um revisão sistemática da literatura será projetada para verificar o estado da arte no contexto do problema de contratos inteligentes e simulação. Por fim, uma pesquisa com especialistas do domínio de contratos inteligentes ajudaria a delinear melhor o problema e a necessidade de solução usando simulação de contratos inteligentes em blockchain.

Referências

Abdellatif, T. e Brousmiche, K.-L. (2018). Formal verification of smart contracts based on users and blockchain behaviors models. In *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–5.

Abreu, A., Coutinho, E. F., e Bezerra, C. I. M. (2022). Performance evaluation of data transactions in blockchain. *IEEE Latin America Transactions*, 20(3):409–416.

Abreu, A. W. S., Coutinho, E. F., e Bezerra, C. I. M. (2020). A blockchain-based architecture for query and registration of student degree certificates. In *Proceedings of the 14th Brazilian Symposium on Software Components, Architectures, and Reuse*, SBCARS '20, page 151–160.

Akca, S., Rajan, A., e Peng, C. (2019). Solanalyser: A framework for analysing and testing smart contracts. In *2019 26th Asia-Pacific Software Engineering Conference (APSEC)*, pages 482–489.

Alharby, M. e van Moorsel, A. (2017). Blockchain-based smart contracts: A systematic mapping study. *arXiv preprint arXiv:1710.06372*.

Bhaskar, N. D. e Chuen, D. L. K. (2015). Chapter 3 - bitcoin mining technology. In Chuen, D. L. K., editor, *Handbook of Digital Currency*, pages 45 – 65. Academic Press, San Diego.

Chapman, P., Xu, D., Deng, L., e Xiong, Y. (2019). Deviant: A mutation testing tool for solidity smart contracts. In *2019 IEEE International Conference on Blockchain (Blockchain)*, pages 319–324.

Christidis, K. e Devetsikiotis, M. (2016). Blockchains and smart contracts for the internet of things. *IEEE Access*, 4:2292–2303.

Coutinho, E. F., Maia, D. J. H., Bezerra, W. L. B., e dos Santos Abreu, A. W. (2020). Avaliando o custo de contratos inteligentes em aplicações blockchain por meio de ambientes de simulação. In *Anais do II Workshop em Modelagem e Simulação de Sistemas Intensivos em Software*, pages 56–65, Porto Alegre, RS, Brasil. SBC.

Destefanis, G., Marchesi, M., Ortu, M., Tonelli, R., Bracciali, A., e Hierons, R. (2018). Smart contracts vulnerabilities: a call for blockchain software engineering? In *2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*.

Liu, C., Liu, H., Cao, Z., Chen, Z., Chen, B., e Roscoe, B. (2018). Reguard: Finding reentrancy bugs in smart contracts. In *2018 IEEE/ACM 40th International Conference on Software Engineering: Companion (ICSE-Companion)*, pages 65–68.

Liu, Y., Li, Y., Lin, S.-W., e Yan, Q. (2020). Modcon: A model-based testing platform for smart contracts. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2020, page 1601–1605.

Nascimento Gomes, A. e Coutinho, E. F. (2021). An Architecture Proposal for E-health Data Collection and Storage Based on Internet of Things and Blockchain. In *9th International Workshop on ADVANCEs in ICT Infrastructures and Services*.

Nofer, M., Gomber, P., Hinz, O., e Schiereck, D. (2017). Blockchain. *Business & Information Systems Engineering*, 59(3):183–187.

Parizi, R. M., Dehghantanha, A., Choo, K.-K. R., e Singh, A. (2018). Empirical vulnerability analysis of automated smart contracts security testing on blockchains. In *Proceedings of the 28th Annual International Conference on Computer Science and Software Engineering*, CASCON '18, page 103–113, USA. IBM Corp.

Singh, A., Parizi, R. M., Zhang, Q., Choo, K.-K. R., e Dehghantanha, A. (2020). Blockchain smart contracts formalization: Approaches and challenges to address vulnerabilities. *Computers & Security*, 88:101654.

Swan, M. (2015). *Blockchain: Blueprint for a new economy*. "O'Reilly Media, Inc.".

Szabo, N. (1994). Smart contracts. http://bit.ly/2Yc9vjb. Online; accessed Oct-2019.

Thakkar, P., Nathan, S., e Viswanathan, B. (2018). Performance benchmarking and optimizing hyperledger fabric blockchain platform. In *2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*.

Xu, X., Pautasso, C., Zhu, L., Gramoli, V., Ponomarev, A., Tran, A. B., e Chen, S. (2016). The blockchain as a software connector. In *2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA)*, pages 182–191.

Zheng, Z., Xie, S., Dai, H.-N., Chen, X., e Wang, H. (2017). Blockchain challenges and opportunities: A survey. *International Journal of Web and Grid Services (IJWGS)*.