



UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIAS DA COMPUTAÇÃO

Marina Luiza Lardizabal Vieira

Representation of Smart Contracts as State Diagrams

Florianópolis
2022

Marina Luiza Lardizabal Vieira

Representation of Smart Contracts as State Diagrams

Dissertação submetida ao Programa de Pós-Graduação em Ciências da Computação da Universidade Federal de Santa Catarina para a obtenção do título de mestre em Ciências da Computação.

Supervisor:: Profa. Patricia Vilain, Dra.

Florianópolis

2022

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Vieira, Marina Luiza Lardizábal Vieira
Representation of Smart Contracts as State Diagrams /
Marina Luiza Lardizábal Vieira Vieira ; orientadora,
Patricia Vilain , 2022.
138 p.

Dissertação (mestrado) - Universidade Federal de Santa
Catarina, Centro Tecnológico, Programa de Pós-Graduação em
Ciência da Computação, Florianópolis, 2022.

Inclui referências.

1. Ciência da Computação. 2. Software Engineering . 3.
Smart Contracts. 4. State Diagram. I. , Patricia Vilain.
II. Universidade Federal de Santa Catarina. Programa de Pós
Graduação em Ciência da Computação. III. Título.

Marina Luiza Lardizabal Vieira

Representation of Smart Contracts as State Diagrams

O presente trabalho em nível de mestrado foi avaliado e aprovado por banca examinadora composta pelos seguintes membros:

Profa. Fabiane Benitti, Dra.
Universidade Federal de Santa Catarina

Prof. Jean Hauck, Dr.
Universidade Federal de Santa Catarina

Prof. Raul Sidnei Wazlawick, Dr.
Universidade Federal de Santa Catarina

Certificamos que esta é a **versão original e final** do trabalho de conclusão que foi julgado adequado para obtenção do título de mestre em Ciências da Computação.

Coordenação do Programa de
Pós-Graduação

Profa. Patricia Vilain, Dra.
Supervisor:

Florianópolis, 2022.

This work is dedicated to my beloved parents,
grandparents and my dear brother.

ACKNOWLEDGEMENTS

First of all, I would like to thank my advisor, who guided me through every step of the way with the research and the writing of this work. I would also like to thank my professional colleagues, who also helped and encouraged me to move forward with the research. Finally, I thank my family and my fiancé, who have always believed in me and who, in their own way, helped me throughout this work.

RESUMO

Smart contracts ganharam popularidade recentemente com o surgimento das *blockchains*, embora o conceito por trás do termo *smart contract* tenha sido estudado desde os anos 90. A automação de contratos firmados na vida real é um assunto interdisciplinar e chama a atenção não só no escopo da tecnologia, mas também em áreas como negócios e direito. Em contraste com um contrato legal escrito em linguagem natural, entender com um *smart contract* funciona pode ser uma tarefa difícil, especialmente para pessoas que não são programadoras. Em contrapartida, contratos escritos em linguagem natural podem conter ambiguidades e muita informação desnecessária, além de um vocabulário complexo e complicado. Com o objetivo de permitir entendimento claro, precisão e segurança das informações no processo de criação de um contrato inteligente, diversas ferramentas foram desenvolvidas, tanto para evitar vulnerabilidades quanto para permitir que qualquer pessoa contribua para a escrita de um contrato. Em paralelo, dentro da Engenharia de Software, muitas ferramentas de design visual são utilizadas para garantir a precisão esperada de um sistema. Diante desse cenário e buscando facilitar ainda mais o entendimento geral dos *smart contracts*, este trabalho visa ao mapeamento automático entre uma linguagem formal para especificação de *smart contracts* em um diagrama de estados. Dessa forma, podemos democratizar ainda mais a compreensão dos contratos legais e possibilitar o uso de *smart contracts*. Para isso, primeiramente foi realizada uma revisão sistemática, com o objetivo de encontrar trabalhos que também representassem *smart contracts* utilizando diagramas de estados ou outros recursos visuais similares, mas que também representassem o ciclo de vida do *smart contract*. Nessa revisão sistemática, também buscamos selecionar trabalhos que mencionassem a representação dos aspectos legais de um contrato, ou seja, a representação de obrigações e poderes, direitos e deveres dentro do *smart contract*. Com a revisão sistemática concluída, foi projetado um mapeamento dos passos necessários para transformar uma linguagem de domínio específica para *smart contracts* em um diagrama de estado. Num primeiro momento, o mapeamento também foi especificado para uma linguagem específica chamada Symboleo. Com o projeto em mente, um algoritmo foi implementado, possibilitando o *upload* de um arquivo contendo um contrato definido com a linguagem Symboleo e produzindo um diagrama de estado para esse contrato. Os dois experimentos realizados mostraram que um diagrama de estado gerado com o algoritmo implementado neste trabalho ajudou os participantes a responder perguntas sobre um contrato escrito em linguagem natural.

Palavras chave: diagramas de estado. *smart contracts*. *blockchain*.

RESUMO EXPANDIDO

Introdução

Smart contract é um termo relativamente recente na Computação, mas que ganhou muita atenção depois da popularização da tecnologia de *Blockchain*. *Smart contracts* nada mais são do que um código executável que executa regras para facilitar e fazer cumprir os termos do acordo entre partes não confiáveis para trabalharem juntas. *Blockchains* armazenam esses *smart contracts* e, assim como as transações, eles também são imutáveis, ou seja, depois de publicados dentro de uma *blockchain*, seus códigos não podem mais ser alterados. O uso de *smart contracts*, apesar das dificuldades técnicas de implementação, pode trazer inúmeros benefícios, começando com a substituição do uso de papel adjunto com a possível prevenção de adulteração e falsificação. Mas a grande vantagem vem com uma promessa de garantia da realização dos termos de um contrato sem a necessidade de envolvimento de uma terceira parte confiável responsável pela execução do contrato. Além dos desafios de trazer o cumprimento de contratos para o mundo digital, existem os de possíveis falhas e vulnerabilidades de segurança no desenvolvimento de um *smart contract*. Mesmo assim, autores defendem a utilização de *smart contracts* com o uso de métodos criptográficos, além da utilização de boas práticas de Engenharia de Software, como a utilização de um processo de desenvolvimento claro e práticas de design e notações úteis para a representação de *smart contracts*. Alguns autores afirmam que *smart contract* é também um conceito interdisciplinar que interessa (mas não limitado) a áreas como negócios/finanças e contratos legais. A natureza interdisciplinar é um dos principais motivadores para a definição de linguagens formais para *smart contracts*. Linguagens formais também foram propostas para construir um modelo seguro para *smart contracts* para detectar vulnerabilidades de segurança em nível de design. Mas até que ponto uma linguagem formal pode ser totalmente entendida por um leigo? É possível usar outros tipos de modelagem e técnicas de design junto com especificações formais para facilitar o entendimento de um *smart contract* por um leigo? Pensando em facilitar o entendimento de todos envolvidos, nós vamos gerar um diagrama de estado para representar um *smart contract* especificado em uma linguagem formal. É importante reparar que uma especificação formal facilita a verificação de comportamento correto de um *smart contract* e também possibilita a geração automática de código de *smart contracts*, enquanto um diagrama de estado pode ser usado para facilitar o entendimento desse *smart contract* por um leigo. Como o diagrama de estados é gerado a partir da especificação formal, ele será usado para confirmar que as informações do *smart contract*, descrito em linguagem natural, também estão sendo incluídas na especificação formal que representa esse *smart contract*. Após essa confirmação, o código correspondente ao *smart contract* pode ser gerado automaticamente a partir da especificação formal.

Objetivos

O objetivo geral desse trabalho de mestrado é criar um algoritmo que transforme um contrato escrito com a linguagem de especificação formal *Symbolico* em um diagrama de estados para facilitar o entendimento e compreensão global do comportamento de um contrato. É feito um mapeamento dos aspectos que definem os estados e suas transições para os estados do contrato.

Metodologia

Primeiramente foi realizado um mapeamento sistemático para responder algumas perguntas de pesquisa importantes relacionadas ao uso de diagramas de estado para representar *smart contracts*. Em cima desse mapeamento, foram propostos os possíveis estados de um contrato de um *smart contract*. Um algoritmo foi definido e implementado para mapear um *smart contract* representado com a linguagem Symboleo para o formato de um diagrama de estado. Para mostrar que o diagrama de estado pode facilitar o entendimento desse *smart contract*, foram feitos experimentos através de questionários verificando se as respostas foram mais assertivas e rápidas com o uso do diagrama de estado.

Resultados e Discussões

Foram realizados dois experimentos para mostrar o melhor entendimento de um *smart contract* com diagrama de estado. No primeiro experimento, foram escolhidos seis participantes da área de Ciências da Computação e também da área de Direito. Foram feitas quatro perguntas onde os participantes deveriam consultar apenas um contrato escrito em linguagem natural e depois mais quatro perguntas onde os participantes utilizaram a ajuda do diagrama de estado. Nesse primeiro experimento, os mesmos participantes participaram das duas etapas de perguntas. Apesar disso, as respostas foram mais assertivas e concisas com o diagrama de estado, bem como o tempo foi notavelmente reduzido. Já no segundo experimento foram entrevistadas dez pessoas, também da área de Ciências da Computação e Direito. Dessa vez, cinco pessoas responderam cinco perguntas sem o uso do diagrama de estado, apenas com o contrato escrito em linguagem natural, e outras cinco pessoas responderam as perguntas com a ajuda do diagrama de estado. Assim como o primeiro experimento, podemos notar respostas mais assertivas e concisas com o uso do diagrama de estado, e também tempos de resposta mais reduzidos.

Considerações Finais

O interesse em *smart contracts* vem aumentando muito e chamando a atenção não só de pessoas da área de Tecnologia, mas também de pessoas da área do Direito e de Negócios. Isso fez com que a necessidade de estudos buscando a facilidade do entendimento em relação ao desenvolvimento de *smart contracts* tenha crescido, bem como estudos que buscam garantir a validação das regras desse contrato. Esse trabalho buscou garantir o entendimento de *smart contracts* escritos em linguagens formais, através da geração de diagramas de estado, principalmente por leigos. Os experimentos realizados, que mostraram que o uso de diagramas de estado diminuiu o tempo de resposta e permitiu respostas mais assertivas, mostram que o diagrama de estado pode facilitar esse entendimento. Apesar disso, outros tipos de experimentos, bem como provas formais e melhoria do diagrama de estado, são alguns dos trabalhos futuros propostos nessa dissertação.

Palavras chave: diagramas de estado. *smart contracts*. *blockchain*.

ABSTRACT

Smart contracts have recently gained popularity with the emergence of blockchains, although the concept behind them has been studied since the 1990s. The automation of contracts signed in real life is an interdisciplinary subject and draws attention not only in the scope of technology but also in areas such as business and law. In contrast to a legal contract written in natural language, understanding how a smart contract works may be a difficult task, especially for non-programmers. On the other hand, contracts written in natural language may contain ambiguities and much unnecessary information, in addition to complicated vocabulary. With the aim of providing clear understanding, accuracy and security of information in the process of creating a smart contract, several tools have been developed, both to avoid vulnerabilities and to allow anyone to contribute to the writing of a contract. In parallel, within Software Engineering, many visual design tools are used to ensure the accuracy expected from a system. In view of this scenario and seeking to further facilitate the general understanding of smart contracts, this master's thesis aims at carrying out automatic mapping between a formal smart contract specification language and a state diagram. In doing so, we can further democratize the understanding of legal contracts and enable the use of smart contracts. The two experiments carried out showed that a state diagram generated with the algorithm developed in this study helped participants answer questions about a contract written in natural language.

Keywords: state diagrams. smart contracts. blockchain.

LIST OF FIGURES

Figure 1 – Contract written with Symboleo	17
Figure 2 – Blockchain example	21
Figure 3 – Example of a smart contract inside a blockchain	24
Figure 4 – State diagram example	25
Figure 5 – State diagram of an aircraft component life cycle	26
Figure 6 – Selected articles	30
Figure 7 – Finite state machine for a blind auction	35
Figure 8 – Rights and obligations between participating parties in a state diagram	35
Figure 9 – Clauses in a meat purchase and sale agreement	43
Figure 10 – Meat purchase and sale agreement specified in the Symboleo lan- guage	44
Figure 11 – State diagram generated manually	45
Figure 12 – Generic state diagram	46
Figure 13 – Json Model	49
Figure 14 – Json Model Transition	50
Figure 15 – Json Model Actions	50
Figure 16 – The proposed Symboleo language ontology	52
Figure 17 – Section of the Domain block specified in Symboleo	53
Figure 18 – Contract parametrization in Symboleo	53
Figure 19 – Legal situations defined in Symboleo	54
Figure 20 – Class diagram API	57
Figure 21 – Sequence diagram API	58
Figure 22 – Class Diagram API	59
Figure 23 – Sequence Diagram API	60
Figure 24 – Final JSON	61
Figure 25 – Final JSON transition	62
Figure 26 – Final JSON actions	63
Figure 27 – Frontend Home	65
Figure 28 – Select file in Frontend	65
Figure 29 – File selected in Frontend	65
Figure 30 – State diagram generated in Frontend	66
Figure 31 – Text for state diagram creation with the Mermaid library	66
Figure 32 – Meat sale manually generated state diagram	67
Figure 33 – Meat sale automatically generated state diagram	68
Figure 34 – Tech service manually generated state diagram	68
Figure 35 – Tech service automatically generated state diagram	69
Figure 36 – State diagram of the contract	72

Figure 37 – Participants’ time spent reading the contract and the diagram . . .	79
Figure 38 – State diagram of the legal contract	81
Figure 39 – Participants’ time spent reading the contract and the state diagram for each question	86
Figure 40 – Participants’ total time spent reading the contract and the state diagram	86
Figure 41 – Final json generated by algorithm part 1	118
Figure 42 – Final json generated by algorithm part 2	119
Figure 43 – Final json generated by algorithm part 3	120
Figure 44 – Final json generated by algorithm part 4	121

LIST OF TABLES

Table 1 – Articles selected by title	29
Table 2 – Types of diagrams used in the papers	36
Table 3 – Subdivision of selected articles dealing with smart contract modeling or with legal contract aspects	37
Table 4 – Comparisons between related work	39
Table 5 – Participants’ profile (Experiment 1)	71
Table 6 – Participants’ profile (Experiment 2)	71
Table 7 – Participants’ answers to the first group of questions	73
Table 8 – Participants’ answers to the second group of questions	76
Table 9 – Participants’ feedback on the use of the state diagram	78
Table 10 – Participants’ answers without the state diagram	82
Table 11 – Participants’ answers with the state diagram	84
Table 12 – Participants’ feedback on the use of the state diagram	85
Table 13 – Articles select in the literature review	98

CONTENTS

1	INTRODUCTION	15
1.1	MOTIVATION	15
1.2	PROBLEM	17
1.3	SOLUTION	17
1.4	SCOPE	18
1.5	AIM AND OBJECTIVES	19
1.5.1	Aim	19
1.5.2	Objectives	19
1.6	METHOD	19
1.7	TEXT ORGANIZATION	20
2	BACKGROUND	21
2.1	BLOCKCHAIN	21
2.2	SMART CONTRACTS	23
2.3	STATE MACHINE DIAGRAMS	23
2.4	DOMAIN-SPECIFIC LANGUAGES AND FORMAL METHODS	26
3	LITERATURE REVIEW	28
3.1	FORMULATION OF THE RESEARCH QUESTION	28
3.2	IDENTIFICATION OF RELEVANT LITERATURE	28
3.3	SELECTION OF STUDIES	29
3.4	RESULTS AND DISCUSSION	36
3.4.1	Modeling with state diagram/state machine	37
3.4.2	Review of the legal aspects of the contract	38
3.4.3	Comparison between the main elements	39
3.4.4	Threats to validity	42
4	PROPOSAL	43
4.1	GENERAL MAPPING OF A LEGAL SMART CONTRACT INTO A STATE DIAGRAM	44
4.1.1	Generic state diagram for a smart contract	45
4.1.2	Requirements of a domain language for a smart contract	46
4.1.3	Steps to define a state diagram for a smart contract	47
4.2	MAPPING A SYMBOLEO SMART CONTRACT INTO A STATE DIAGRAM	51
4.2.1	The Symboleo language	51
4.2.2	Detail of the mapping from a Symboleo specification	53
4.3	MAPPING IMPLEMENTATION	56
4.3.1	API Project	57
4.3.2	API implementation	57
4.3.3	Frontend	64

4.3.4	Development evaluation	67
5	EXPERIMENTS	70
5.1	EXPERIMENT PROTOCOL	70
5.2	EXPERIMENT 1	72
5.2.1	Overview	72
5.2.2	Execution	73
5.2.3	Results	79
5.2.4	Threats to validity	80
5.3	EXPERIMENT 2	80
5.3.1	Overview	80
5.3.2	Execution	81
5.3.3	Results	85
5.3.4	Threats to validity	87
6	CONCLUSIONS	89
6.0.1	Future Work	90
	REFERENCES	91
	APPENDIX A – LITERATURE REVIEW – ARTICLES’ TITLES	98
	APPENDIX B – JSON GENERATED WITH THE ALGORITHM	117
	ANNEX A – CONTRACT FOR EXPERIMENT 1	122
	ANNEX B – CONTRACT FOR EXPERIMENT 2	132

1 INTRODUCTION

The term smart contract is relatively recent and gained considerable attention after the popularization of the blockchain technology. A smart contract is nothing but executable code used to facilitate and enforce the contract terms that make sense on the blockchain (ABDELHAMID; HASSAN, 2019). Blockchains store these smart contracts, and, like transactions, smart contracts are also immutable, that is, once published within a blockchain, their codes can no longer be changed (NARAYANAN; MILLER, 2016).

However, the appearance of such term is not that new. Nick Szabo, in 1997, was already studying the possibility of contract automation and discusses how the terms of a contract should be written in lines of code in a self-executing contract (SZABO, 1997). Szabo's definition specifically deals with contracts within the scope of legal aspects and not as programs that execute instructions on a blockchain. However, initially, his theory did not make much progress, mainly due to the difficulty in controlling the physical assets of a contract, i.e., the lack of a technology capable of supporting what he was proposing. With the possibility of implementing a smart contract on a blockchain, which already has its asset under control, such as a digital currency, Szabo's proposal became feasible (BAI et al., 2018). In this master's thesis, we address this kind of smart contracts – the ones that are written to represent real-world legal contracts.

The use of smart contracts, despite technical implementation difficulties, can bring many benefits, such as the possibility of preventing adulteration and forgery (CHANG; LUO; CHEN, Y., 2020). Nevertheless, the biggest advantage comes with a promise of guaranteeing that the terms of a contract does not require the involvement of a trusted third party responsible for executing the contract. Even though in practice many legal aspects of contracts are left out when automating them, some authors argue that a contract can be better designed using technology instead of just a text written in natural language on paper (SKOTNICA; PERGL, 2020). This way, a "large amount of repetitive administrative work can be eliminated, the comprehension can be increased, and third-party involvement reduced, as the courts, for example" (SKOTNICA; PERGL, 2020).

1.1 MOTIVATION

Despite all existing advancements, converting a legal contract into executable code is a challenging task. The vagueness of laws, for example, which embraces the largest number of cases in justice systems, makes it difficult to achieve accuracy when reading through such laws, chiefly when programming an algorithm to verify compliance thereof (SKOTNICA; PERGL, 2020). "The vagueness of law is a required

and important property in justice systems. However, when people or companies need to comply with the laws and design systems that will automate the associated work, they need a clear definition of how does the law apply in their case” (SKOTNICA; PERGL, 2020). Yet, the problem goes beyond the gap between natural language and equivalent code (WOHRER; ZDUN, U., 2020). There are also other issues, such as: the fixed and autonomous nature of code execution in the blockchain environment, since it cannot be changed once deployed; the lack of high-level coding abstractions; and the rapid progress of the development framework (WOHRER; ZDUN, U., 2020).

In addition to the challenges of bringing contract enforcement to the digital world, there are potential security flaws and vulnerabilities in the development of a smart contract, described as follows: “Firstly, smart contracts deployed in practice handle financial assets of significant value. Secondly, smart contract bugs cannot be patched. By design, once a contract is deployed, its functionality cannot be altered even by its creator. Finally, once a faulty or malicious transaction is recorded, it cannot be removed from the blockchain” (MAVRIDOU; LASZKA, 2018).

Despite inherent difficulties, Landleif (LADLEIF; WESKE, 2019b) argues that there are two reasons for believing in the creation and use of smart contracts: the validity of real-world contracts can be secured via cryptographic methods, and the fact that they contain operational aspects which can be automated by computers. In addition to these factors, the use of good software engineering practices, with a clear development process, and the use of design practices and notations useful for the purpose can prevent various problems from occurring (MARCHESI, M.; MARCHESI, L.; TONELLI, 2018). The use of a systematic design may be able to avoid a number of failures and wrong development assumptions, because, as stated by Luu (LUU et al., 2016): “Flaws arise in practice because of a semantic gap between the assumptions contract writers make about the underlying execution semantics and the actual semantics of the smart contract system”.

One of the main alternatives to overcome difficulties is the use of formal languages. Formal languages have been proposed to “build secure models for smart contracts to detect security vulnerabilities at the design level” (XU; FINK, 2020). The interdisciplinary nature thereof is another motivator for the definition of formal languages for smart contracts. According to He (HE, Xiao et al., 2018) “Although a smart contract is usually viewed as an online program from the perspective of information technology, it is actually an interdisciplinary concept that also concerns (but is not limited to) business/finance and contract law”. The interdisciplinary nature of smart contracts is one of the main motivators for the definition of formal languages, which, according to said author, can facilitate the understanding of technology laypeople in the process of creating a smart contact (HE, Xiao et al., 2018).

1.2 PROBLEM

Formal languages can be used to specify smart contracts and also to generate the code corresponding to a smart contract, as happens with Symboleo (SHARIFI, S. et al., 2020). But to what extent can a formal language be fully understood by a layperson? Is it possible to use other modeling and design techniques along with a formal specification to facilitate the understanding of a smart contract by a layperson?

Figure 1 shows a contract written with Symboleo (SHARIFI, S. et al., 2020) whose understanding, as can be seen, may not be as typical as that of a contract written in natural language.

Figure 1 – Contract written with Symboleo

Table 6 Symboleo specification of the transactive energy (TE) contract.

<pre> Domain transactiveEnergyAgreementD ISO isA Role; DERP isA Role; DispatchInstruction isA Asset with maxVoltage: Integer, minVoltage: Integer; Bid isA Asset with id: idCode, by: DERP, dispatchHour: Integer, energy: Integer, price: Integer, instruction: DispatchInstruction; BidAccepted isA Event with bid: Bid; EnergySupplied isA Event with energy: Integer, dispatchHour: Integer, by: DERP, voltage: Integer, ampere: Integer; Invoice isA Asset with id: idCode, date: Date, price: Integer; InvoiceIssued isA Event with issuedInvoice: Invoice; NoticeIssued isA Event with date: Date; Paid isA Event with invoice: Invoice, from: Role, to: Role; endDomain Contract transactiveEnergyAgreementC(caiso: ISO, derp: DERP) Declarations bid: Bid; bidAccepted: BidAccepted; energySupplied: EnergySupplied; terminationNoticeIssued: NoticeIssued; isoPaid: Paid; supPaid: Paid; creditInvoiceIssued: InvoiceIssued; Preconditions Postconditions Obligations O_{payByISO} : happens(creditInvoiceIssued, t) → O(caiso, derp, true, happensWithin(isoPaid, t + 4 days)); O_{supplyEnergy} : happens(bidAccepted, t) → O(derp, caiso, true, happens(energySupplied, bidAccepted.bid.dispatchHour)); O_{issueInvoice} : happens(exerted(P_{imposePenalty}.instance), t) → O(derp, caiso, true, happens(supPaid, t + 4)); SurvivingObls Powers P_{terminateAgreement}: P(caiso, derp, violates(O_{issueInvoice}.instance) and happensBefore(terminationNoticeIssued, now.time-30), terminates(self)); P_{terminateAgreementBySupplier}: P(derp, caiso, happensBefore(terminationNoticeIssued, now.time-90), terminates(self)); P_{imposePenalty}: violates(O_{supplyEnergy}.instance) → P(caiso, derp, true, creates(O_{issueInvoice})); Constraints not(isEqual(buyer, seller)); endContract </pre>

1.3 SOLUTION

A smart contract is an agreement or set of rules that govern a business transaction and is executed automatically as part of a transaction, being able to automate

sensitive details of a negotiation between two parties. For example, a smart contract may define contractual conditions under which corporate bond transfer occurs (MANAV, 2017). It is vital that all parties of a contract be aware of the purpose of the agreement, i.e., the terms and conditions of the contract, its stages, and what actions can change its states, for instance, actions that can make the contract move from a status of being in effect to duly concluded.

This way, seeking to facilitate the understanding of smart contracts for all people involved in creating them, mainly the users requesting the contract (lawyers and their clients) and also the analysts, this study aims to represent a contract written with a domain-specific language for smart contracts as a UML state diagram. The UML language is a standard, it has a graphical and intuitive representation, and it has an extensibility for representing domain-specific notations (BARESI; GARZOTTO; PAOLINI, 2001). Concerning a state diagram specifically, “beyond their usual purpose of making explicit the state through which an element evolves, they become a means of reasoning about some peculiar aspects of application and about time constraints” (BARESI; GARZOTTO; PAOLINI, 2001).

States and actions that prompt changes from one state to another in a contract can facilitate the understanding and development of business rules. In this case, we would be dealing with the possible states of a smart contract, which in turn represents a legal contract. Researcher Jeff Edmonds, for example, in his book “How to Think About Algorithms”, recommends a paradigm that seeks to facilitate the understanding of algorithms as a sequence of states, in addition to a (often very large) sequence of steps (EDMONDS, 2008). Thus, effectively modeling the states of a smart contract is pointedly necessary as it is nothing more than an algorithm.

In the present study, to allow for further understanding, we will generate a state diagram to represent a smart contract specified with a formal domain language. It is worth noting that a formal specification facilitates verifying whether a smart contract is correctly functioning and allows the generation of the smart contract code, whereas a state diagram can be used to facilitate its understanding by a layperson. Since the state diagram is generated from the formal specification, it will be used to confirm that what is written in the legal contract represented in natural language is also being included in the formal specification representing the smart contract. After this confirmation, the code corresponding to the smart contract can be automatically generated from the formal specification.

1.4 SCOPE

The smart contracts used in this study are those to represent legal contracts. The formal language used is Symboleo (SHARIFI, S. et al., 2020). The choice of this specific language is due to the easy access to its documentation, in addition to the

contact with the group of study involved in the language development.

1.5 AIM AND OBJECTIVES

In this section, the aim and objectives of the study will be presented.

1.5.1 Aim

The aim of this master's thesis is to map a smart contract written with the domain-specific language Symboleo (SHARIFI, S. et al., 2020) and represent it into a state diagram that can facilitate an overall understanding of the contract's behavior, therefore aspects defining the states of the contract and its transitions will be mapped.

1.5.2 Objectives

- Specify an algorithm – a step-by-step specification of what this algorithm needs to do, specifying the input data (as regards a contract written with the Symboleo language) and what should be the output (as regards a state diagram);
- Implement the specified algorithm – according to the defined specification, implement an algorithm that can receive the specified inputs and compute the expected output;
- Validate the specified algorithm through experiments, checking whether the state diagram facilitates the understanding of the contract life cycle through interviews with experts in law who are unknowledgeable about computing and vice versa.

1.6 METHOD

1. Characterization of the study
 - a) Approach: qualitative research;
 - b) Technical procedures: bibliographic research, experimental research including a case study.
2. Methodological procedures
 - a) Step 1: Systematic mapping with representations of smart contracts and ways of representing their states in diagrams;
 - b) Step 2: Study of the possible states (life cycle) of smart contracts inside a blockchain – Created, In effect, Suspended, Concluded, etc.;

- c) Step 3: Definition and implementation of an algorithm to map a Symboleo formal model into a state diagram;
- d) Step 4: Experiments carried out through questionnaires to assess whether the responses thereof are more assertive with the use of the state diagram generated with the algorithm of Step 3;
- e) Step 5: Writing an article about the work developed;
- f) Step 6: Writing the thesis and presenting it.

1.7 TEXT ORGANIZATION

The subsequent sections are organized according to the chapters described as follows. Chapter 2 (Background) introduces topics that are important for the understanding of this study, such as blockchain, smart contracts, and state diagrams. Chapter 3 (Literature Review) presents related work that helped review the state of the art for this study. Chapter 4 (Proposal) describes the proposal of the present study, followed by its implementation. Chapter 5 (Experiments) presents two experiments that were performed to verify the hypothesis put forward. Finally, Chapter 6 (Conclusion) draws conclusions based on the experiment and execution of the proposal.

2 BACKGROUND

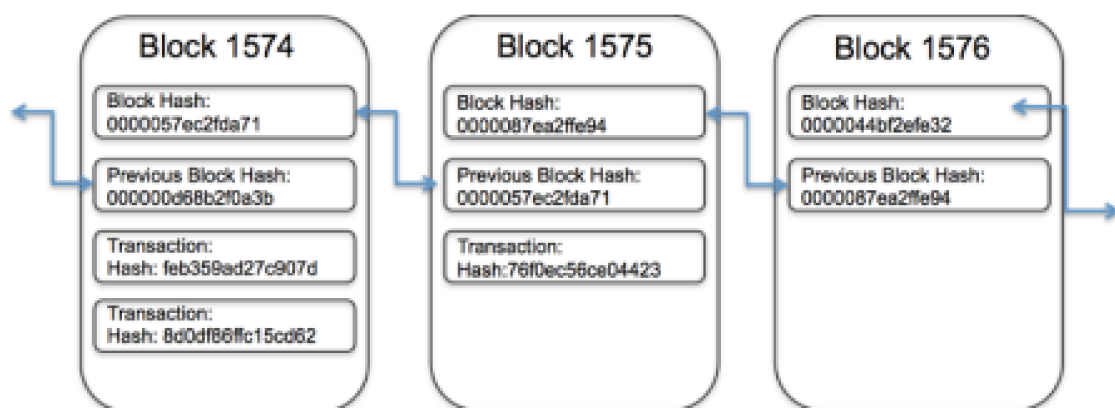
The following sections will discuss some topics that are crucial to the understanding of the context of this master's thesis, comprising the definition of blockchains, smart contracts, and state diagrams.

2.1 BLOCKCHAIN

A blockchain system, as conceptualized by (IDELBERGER et al., 2016), "consists of a network of computing nodes, sharing a common data structure (the blockchain) with consensus about the state of this structure". The term emerged in 2008 along with the creation of Bitcoin, which was created with the objective of allowing the transfer of payments from one party to another without the need for intermediaries to ensure security, using, for this, cryptographic and authentication operations (GATTESCHI et al., 2018). In the case of cryptocurrencies, such as Bitcoin, a blockchain works as an accounting book where all accounts (wallets) and transfers are recorded (IDELBERGER et al., 2016). It is worth noting that, in addition to being public, once recorded, the information cannot be modified or deleted, that is, it is immutable (GATTESCHI et al., 2018).

Figure 2 shows an example of how a blockchain works. In fact, the word itself is self-explanatory, meaning a data structure formed by blocks that are linked to each other, thus forming a chain. Each block contains a unique identifier in hash format, timestamped batches of recent valid transactions, and the hash of the previous block. The previous block hash links the blocks together and prevents any block from being altered or a block being inserted between two existing blocks (LAURENCE, 2019).

Figure 2 – Blockchain example



Source: (LAURENCE, 2019)

But there needs to be a consensus on the state of the structure, and for that to happen, there is a great deal of processing behind simply performing transactions. Consensus is a process of agreement among distrusting nodes on a final state of data (BASHIR, 2017). For consensus to exist, some nodes actively work through mining. “During mining, nodes check previous transactions to verify whether a subject is entitled to spend a given amount of cryptocurrency and, each time a block has to be added to the chain, solve a complex computational-intensive mathematical problem” (GATTESCHI et al., 2018).

There are various requirements which must be met in order to provide the desired results in a consensus mechanism. Imran Bashir (BASHIR, 2017) described them in his book:

- **Agreement:** All honest nodes decide on the same value;
- **Termination:** All honest nodes terminate execution of the consensus process and eventually reach a decision;
- **Validity:** The value agreed upon all honest nodes must be the same as the initial value proposed by at least one honest node;
- **Fault tolerant:** The consensus algorithm should be able to run in the presence of faulty or malicious nodes (Byzantine nodes);
- **Integrity:** No node makes a decision more than once. The nodes make decisions only once in a single consensus cycle.

There are various types of consensus mechanism, but the two most common types are (BASHIR, 2017):

- Byzantine fault tolerance-based: With no compute intensive operations such as partial hash inversion, this method relies on a simple scheme of nodes that are publishing signed messages. Eventually, when a certain number of messages are received, then an agreement is reached;
- Leader-based consensus mechanisms: This type of mechanism requires nodes to compete for the leader-election lottery and the node that wins it proposes a final value.

For each type of mechanism, there are some algorithms available. The most well-known one, which is implemented in Bitcoin as well as in other cryptocurrencies, is the algorithm called “Proof of Work”: This type of consensus is based on the proof that sufficient computational resources have been allocated before proposing a value for acceptance by the network (BASHIR, 2017).

2.2 SMART CONTRACTS

Smart contracts are simply executable code that executes rules to facilitate and enforce the terms of an agreement between untrusted parties to work together (ABDELHAMID; HASSAN, 2019). As previously mentioned, blockchains store these smart contracts and, like transactions, they are also immutable, that is, after they are published within a blockchain, their codes can no longer be changed (NARAYANAN; MILLER, 2016). Smart contracts can rely on the use of external services (called “oracles”) to ensure their conditions with “real-world” data (death records, for example) and push them to the blockchain or vice versa (GATTESCHI et al., 2018).

Figure 3 illustrates a smart contract within a blockchain. Smart contracts help information to be updated consistently. A blockchain uses smart contracts to provide controlled access to the ledger (HYPERLEDGER, 2020), that is, prior to any access or insertion into the blockchain, a smart contract “runs” its rules and can perform any action that is required before proceeding with the change or reading in the blockchain.

Business transactions are conducted within the guidelines of a legal contract, and the combination of IoT (Internet of Things) and DLT (Distributed Ledger Technology) platforms has offered an unprecedented ability to automate the monitoring of legal contracts, especially for supply chain provenance (SHARIFI, S. S., 2020). Therefore, it is also possible to write smart contracts able to verify whether such guidelines are being complied with. It is precisely these types of smart contracts that we sought to address in this master’s thesis.

As Gatteschi (GATTESCHI et al., 2018) exemplified: “With a smart contract, a person could, for instance, encode his/her will in the blockchain in the form of a set of rules. In case of death, the smart contract could then automatically transfer the testator’s money or other kind of assets to the beneficiary.”

2.3 STATE MACHINE DIAGRAMS

As explained in (SOURCEMAKING, 2007), “Persons, objects, or concepts from the real world, which we model as objects in the IT system, have “lives””. Although life in the real world and its representation as a software object might have differences, both will have birth and death, for example. Inside an IT system, while the object is alive it will be read and changed. These changes might not be subject to any restrictions, but as soon as rules for modification become dynamic, it is then important to document these rules somewhere. In summary, in certain cases, it should be possible to determine whether an event is permitted in the current state of the object and how the object will react to the event (SOURCEMAKING, 2007). One good way to document this behavior is using state machine diagrams.

As defined by OMG (Object Management Group), a state machine diagram,

Figure 3 – Example of a smart contract inside a blockchain

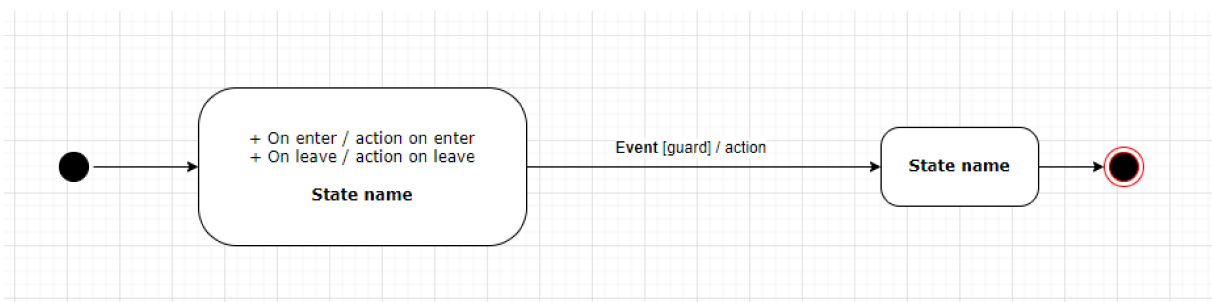


Source: (HYPERLEDGER, 2020)

or simply a state diagram, is “used for modeling discrete event-driven behaviors using a finite state-machine formalism” (OMG, 2017). State machine diagrams can be divided into behavior state machine and protocol state machine. The first type is used to define the behavior of the system, or parts of it, while the second type is used to specify valid interaction sequences, which are called a protocol, hence the name (OMG, 2017).

A state diagram is composed mainly of states and transitions, which are represented by nodes and arcs, respectively. The Figure 4 illustrates a simple state diagram with the main elements. The states are represented as nodes and each node has a name describing its state. The state nodes may contain more information like the actions that happen on entering or leaving each state. The arcs represents the transitions from one state to another and contain the events that trigger such transitions. A transition may contain a condition to happen (guard) and actions that are triggered when the event associated to the transition happen.

Figure 4 – State diagram example

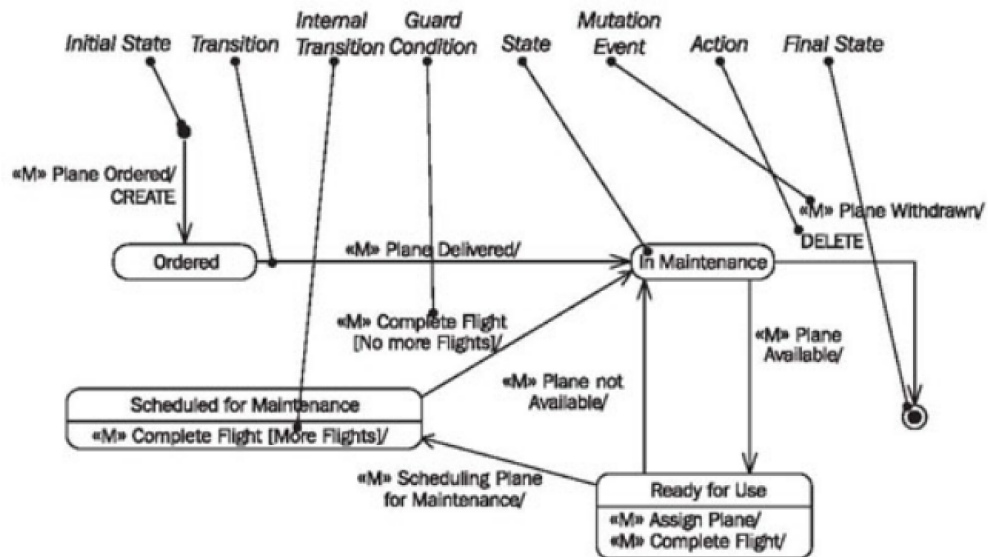


The diagram can also contain other components, like regions, and each component has a lot of specific rules, concepts and subtypes. However, to understand a state diagram in the context of the present study, it suffices to know the following definitions of state and transition:

- State: “A state models a situation in the execution of a state machine behavior during which some invariant condition holds” (OMG, 2017)
- Transition: “A transition is a single directed arc originating from a single source state and terminating on a single target state” (OMG, 2017). A transition may own a set of triggers, each of which specifies an event whose occurrence, when dispatched, may trigger traversal of the transition (OMG, 2017).

Figure 5 shows a state diagram of an aircraft component life cycle after manufacture. The illustration also points to all main elements of a state diagram, such as the states and the aforementioned transitions, as well as other elements that can further detail the life cycle of an object.

Figure 5 – State diagram of an aircraft component life cycle



Source: (SOURCEMAKING, 2007)

2.4 DOMAIN-SPECIFIC LANGUAGES AND FORMAL METHODS

During the last decade, software correctness has been an important and well-studied issue. Many approaches have been proposed and many tools have been implemented to support this quest for correctness. One approach is the use of Domain Specific Languages (DSLs) (BODEVEIX et al., 2005).

A DSL contains domain-specific abstractions as well as domain-specific restrictions that enable verification of domain-specific properties (BODEVEIX et al., 2005). DSLs trade generality for expressiveness in a limited domain. They provide notations and constructs tailored toward a particular application domain, thus offering substantial gains in expressiveness and ease of use compared with general-purpose programming languages (GLPs), with gains in productivity and reduced maintenance costs (MERNIK; HEERING; SLOANE, 2005).

Another approach is the use of formal methods. Such methods associate mathematically rigorous proofs with each step in software design and development (BODEVEIX et al., 2005) or, according to (BAI et al., 2018), they are mathematics-based techniques that describe system attributes for the specification, development, and validation of software. With the use of formal methods in software design, it is expected that appropriate mathematical analyses be performed towards the reliability and robustness of a design (BAI et al., 2018).

The use of these approaches is common in smart contracts because such

contracts need to be reliable and, as they are executable code, they need to meet the following requirements (BAI et al., 2018): 1) cannot have grammatical errors or semantic errors; 2) have high requirements regarding correctness and other related properties to ensure the security of their assertions.

3 LITERATURE REVIEW

Considering that the aim of this study is to map a formal specification language of a legal contract into a state diagram, the first step taken was a systematic mapping review of the literature to find related work that could represent a smart contract in the form of a state diagram. A systematic mapping study enables an overview of a specific research area, indicating which topics have already been addressed in the literature (PETERSEN; VAKKALANKA; KUZNIARZ, 2015). In this study, systematic mapping was conducted according to the guide presented in (PETERSEN et al., 2008) and updated in (PETERSEN; VAKKALANKA; KUZNIARZ, 2015).

As previously mentioned, our purpose was to search for different representations of smart contracts in the form of a state diagram, even though these representations were not our main focus. The intention was to verify how states and transitions were represented in related work. Additionally, we verified whether all related work defined state diagrams based on a formal specification language used to specify smart contracts, i.e., to some extent similarly to the aim of this study.

The next subsections explain the phases of the research: formulation of the research question, identification of relevant literature, selection of studies, extraction of information and, finally, discussion and results of the related work selected.

3.1 FORMULATION OF THE RESEARCH QUESTION

The purpose of this mapping is to understand how smart contracts can be represented as state diagrams, focusing on which states are identified and which transitions connecting such states are identified. Therefore, the main research question is as follows:

- How are transitions and states of a smart contract identified?

In addition to the main question, other two important questions were formulated:

- Is there any standard to name states and transitions of a smart contract state diagram?
- What are the common information sources for creating a smart contract state diagram? (contract in natural language, diagrams, contract represented in formal language, etc.)

3.2 IDENTIFICATION OF RELEVANT LITERATURE

With the aim of finding state diagrams that could represent a smart contract, a search string was created by joining both terms – smart contract and state diagram. Additionally, to further expand the search, some synonymous terms that also

represent states and transitions were used. Thus, the search string was defined as follows:

("state diagram" OR "state machine" OR "statechart" OR "state transition diagram") AND ("smart contract").

The search was conducted on the following academic search engines: ACM Digital Library, IEEExplore Digital Library, Scopus, Springer, and Google Scholar. The first search was conducted in November 2020. Another search was conducted on August 2022 with the aim of updating this mapping with more recent studies, filtering papers from 2021 until 2022.

3.3 SELECTION OF STUDIES

To select the relevant studies, the following inclusion criteria were used: The text must (1) be written in English, (2) be available on the Internet, and (3) have at least one representation of a smart contract in the diagram format where it highlights its states and transitions. Exclusion criteria were applied if: (1) Not written in English, (2) not available on the Internet, and (3) not having any representation of a smart contract in diagram or having a representation not specifically of a smart contract, i.e., just a simple set of business rules that do not represent a contract.

After defining the search string and the inclusion criteria, the search was carried out on the abovementioned search engines. Figure 6 shows the steps followed in the search and the number of studies found in each step, separated by search engine. The first step consists in consulting those digital libraries using the aforementioned search string, adapted to each search engine. The second step is the initial phase of data screening, by reading the titles and selecting the publications most aligned with this study. The third step consisted of removing duplicate titles, that is, those found in more than one digital library, and proceeding with the second phase of data screening, by reading the abstracts of the publications. The fourth step implements one more screening, by reading the introduction and conclusion sections of the publications. Finally, the fifth step is the full reading of the selected publications. All the titles found in the second, third, fourth and fifth steps are listed in Table 13 of Appendix A. The JabRef¹ tool was used to manage all the articles found in each search step.

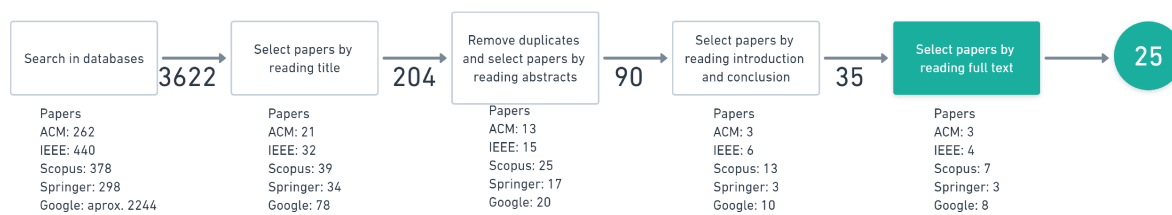
The search results considered relevant to this study can be grouped according to two factors: (1) those addressing the modeling of a smart contract with state diagrams and (2) those dealing with the legal attributes of a contract.

Table 1 lists the 26 selected titles and a brief description of each one.

Table 1 – Articles selected by title

¹ <http://www.jabref.org/>

Figure 6 – Selected articles



#	Title
1	<p>An Agile Software Engineering Method to Design Blockchain Applications (MARCHESI, M.; MARCHESI, L.; TONELLI, 2018)</p> <p>It defines a step-by-step design of blockchain-oriented software (BOS) using agile methodology, defining objectives, identifying actors, and creating UML diagrams. It uses class diagrams, user stories, and even state diagrams.</p>
2	<p>Formal Modeling and Verification of Smart Contracts (BAI et al., 2018)</p> <p>For security and attack prevention, it applies formal method to smart contracts and a general template for smart contract is given, extending this definition to a state machine and using existing tools to verify its formal description.</p>
3	<p>On Legal Contracts, Imperative and Declarative Smart Contracts, and Blockchain Systems (GOVERNATORI et al., 2018)</p> <p>It focuses on a comparison between declarative and imperative language for programming a smart contract. The author states that a smart contract can represent a legal contract taking into account legal aspects. The life cycle of a legal contract is also defined, turning it into a state machine.</p>
4	<p>Towards Model-Driven Engineering of Smart Contracts for Cyber-Physical Systems (GARAMVÖLGYI et al., 2018)</p> <p>It creates a smart contract for Cyber Physical Systems using the model-driven development methodology using UML statecharts, that is, before programming the smart contract, its modeling is detailed in states and transitions.</p>
5	<p>Formal Verification of Smart Contracts Using Interface Automata (MADL et al., 2019)</p> <p>It creates a formal verification method for smart contracts and uses a system of loyalty points to exemplify the method. An automaton is modeled to represent its transitions. Verification techniques are presented to check if the automaton is correct and compatible.</p>
6	<p>Building Executable Secure Design Models for Smart Contracts with Formal Methods (XU; FINK, 2020)</p> <p>It proposes a modeling of a state machine where – using the design-by-contract concern – seeks to avoid future failures and vulnerabilities in modeling the “rights and obligations” of the contract, using the “Temporal Logic of Actions (TLA)” security model.</p>

#	Title
7	<p>Verification-Led Smart Contracts (BANACH, 2020)</p> <p>It focuses on a top-down modeling approach to smart contracts. Among the refinement steps is the creation of a finite state machine to represent the smart contract.</p>
8	<p>Das Contract - A Visual Domain Specific Language for Modeling Blockchain Smart Contracts (SKOTNICA; PERGL, 2020)</p> <p>It uses Business Process Modeling Notation (BPMN) to create diagrams for modeling a contract, and also a diagram to define the process flow – Contract Process Diagram – which aims to create a visualization of the contract flow, respecting its transitions.</p>
9	<p>VeriSolid: Correct-by-Design Smart Contracts for Ethereum (MAVRIDOU et al., 2019)</p> <p>The VeriSolid application is an improvement on the FSolidM tool, with a focus on generating the smart contract from fail-safe modeling (correct-by-design), also derived from a state machine, but with the creation of a verification process of the generated models.</p>
10	<p>Designing Secure Ethereum Smart Contracts: A Finite State Machine Based Approach (MAVRIDOU; LASZKA, 2018)</p> <p>It aims to create a tool that, from the specification of a smart contract in the format of a finite state machine, allows to automatically transcribe it into smart contract code for the Ethereum platform, written with the Solidity language.</p>
11	<p>A Legal Interpretation of Choreography Models (LADLEIF; WESKE, 2019a)</p> <p>Obligations and powers and all legal interaction between parts of a contract are specified as choreography, where the author even shows relationships between the interactions and the legal states of a contract.</p>
12	<p>A Blockchain-Based Decentralized System for Proper Handling of Temporary Employment Contracts (PINNA; IBBA, 2018)</p> <p>It proposes a blockchain application for temporary employee contracts. It creates smart contracts of the employment relationship as a system of states that describe each phase of the relationship. To represent these states, state diagrams and PetriNets are used.</p>
13	<p>Verifying Smart Contracts with Cubicle (CONCHON; KORNEVA; ZAÏDI, 2020)</p> <p>It focuses on creating a two-tier structure for smart contract verification with the Cubicle model checker. To exemplify and prove the method, a smart contract automaton for an auction system is created.</p>
14	<p>Symboleo: Towards a Specification Language for Legal Contracts (SHARIFI, S. et al., 2020) It proposes a new domain specific language for writing smart contracts considering the parts involved and the powers and obligations of them.</p>

#	Title
15	<p>Smart Contracts Using Blockly: Representing a Purchase Agreement Using a Graphical Programming (WEINGAERTNER et al., 2018)</p> <p>It uses Google's Blockly graphical language to prove that it is possible to simplify the creation of smart contracts, taking into account legal contracts, enabling their creation by non-computer experts. Statecharts are specified in the body of the paper.</p>
16	<p>Formal Verification of Functional Requirements for Smart Contract Compositions in Supply Chain Management Systems (ALQAHTANI et al., 2020)</p> <p>It intends to verify the interaction between smart contracts in a supply chain system. In order to verify the integration, state machines are created for each smart contract modeled for the Blockchain.</p>
17	<p>A Smart Contracting Framework for Aggregators of Demand-Side Response (ELIZONDO et al., 2019)</p> <p>It proposes a framework to facilitate the integration of operational flexibility of distributed energy resources (DER) into balancing services for electricity systems. A contribution of the framework is the specification of finite machines of logic states and contract execution.</p>
18	<p>A Model-Driven Approach to Smart Contract Development (BOOGAARD, 2018)</p> <p>It is a master's thesis that proposes a new approach to model-driven engineering where the designing of state diagrams is also part of this modeling.</p>
19	<p>Modeling and Analyzing Smart Contracts using Predicate Transition Nets (HE, Xudong, n.d.)</p> <p>It proposes a modeling and analysis of smart contracts through PetriNets, which also specifies states and transitions. The proposal is exemplified with a blind auction contract.</p>
20	<p>Enforcing commitments with blockchain: an approach to generate smart contracts for choreographed business processes (BERTOLINI, 2020)</p> <p>It is a master's thesis that aims to design a business model with multiple organizations by modeling commitments using choreographies (BPMN) to be able to transform them into smart contracts.</p>
21	<p>Blockchain Medicine Administration Records (BMAR): Reflections and Modelling Blockchain with UML (MITCHELL, n.d.)</p> <p>It aims to model a blockchain application for a Medicines Administration Records (MAR) system to prove that it is valid to build a blockchain for such a system. The author creates a model using UML diagrams, including the state machine diagram.</p>

#	Title
22	<p>SPESC-Translator: Towards Automatically Smart Legal Contract Conversion for Blockchain-based Auction Services (CHEN, E. et al., 2021)</p> <p>It aims to design conversion rules from contracts written in advanced smart contract languages (DSL) to the programming language Solidity.</p>
23	<p>A Model for Verification and Validation of Law Compliance of Smart Contracts in IoT Environment (AMATO et al., 2021)</p> <p>It proposes a formal model - multiagent logic and ontological description of contracts - for validating law compliance of smart contracts and to determine potential responsibilities of failures.</p>
24	<p>Contract as automaton: representing a simple financial agreement in computational form (FLOOD; GOODENOUGH, 2021)</p> <p>It aims to show that fundamental legal structure of a financial contract follows a state-transition logic that can be formalized mathematically as a finite-state machine .</p>
25	<p>Solidity Code Generation From UML State Machines in Model-Driven Smart Contract Development (JURGELAITIS; CEPONIENE; BUTKIENE, 2022)</p> <p>It aims to model a blockchain smart contract with a structured approach based on the model driven architecture. The author creates a model using UML class diagrams and the state machine diagram.</p>

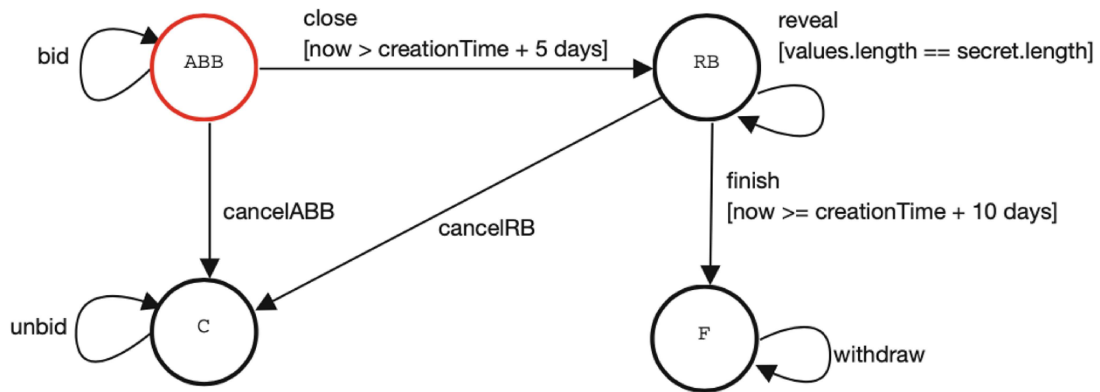
Articles #2, #5, #7, #9, #10, #13, #16, #17, #23 and #24 sought to represent smart contracts using a state machine definition mainly to allow verifying their models by using existing verification models to avoid possible contract vulnerabilities. Articles #1, #4, #18, #21 and #22 use UML state diagrams or statecharts, which are very similar and have the same purpose as state machines. Figure 7, as an example, shows a finite state machine used to represent a blind auction smart contract, where states and transitions were pre-defined to serve as the basis for creating the smart contract (MAVRIDOU; LASZKA, 2018).

Articles #8, #11, #12, #19, #20 and #25 use BPMN process diagrams or PetriNets, either to simplify and facilitate the visualization of the contract life cycle or to guarantee more accurate modeling, also ensuring flawless smart contracts.

In articles #3, #6, #11, #14, #15, #23, #24 and #25, the authors were concerned with correctly covering the behavior of a contract within the legal scope. These articles sought to represent the obligations and powers of the parties involved in a diagram that maps a contract in an executable code. Articles #3, #6, #14, #15, #23, #24 and #25, besides covering a legal smart contract, include contract lifecycle representations using finite-state machines. Figure 8 displays a state diagram that represents the rights and obligations contained in a smart contract extracted from the article by (XU; FINK, 2020), which specifies a property sale agreement.

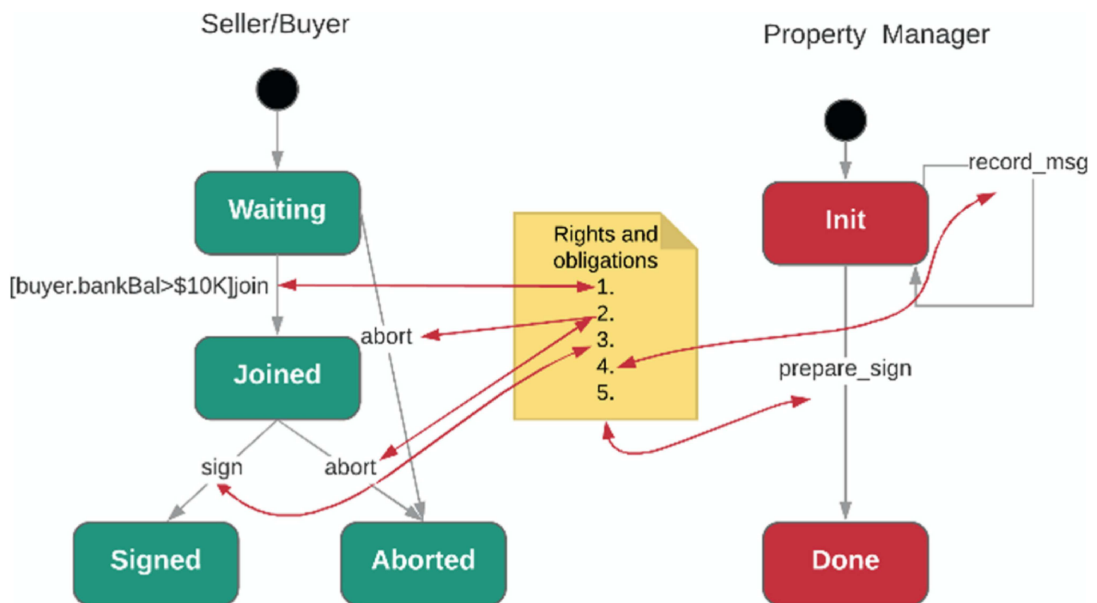
Table 2 summarizes the types of diagrams used per article, where FSM stands

Figure 7 – Finite state machine for a blind auction



Source: (MAVRIDOU; LASZKA, 2018)

Figure 8 – Rights and obligations between participating parties in a state diagram



Source: (XU; FINK, 2020)

for Finite State Machine, PN for PetriNets, BPMN for Business Process Model Notation. LC means Legal Contract and identifies whether the smart contract that was specified in the article represents a legal contract of the real world.

Table 2 – Types of diagrams used in the papers

Types	Articles	Total
FSM	1,2,3,4,5,6,7,9,10,13,14,15,16,17,18,21,22,23,24	19
PN	12,19,25	3
BPMN	8,11,20	3
LC	3,6,11,14,15,23,24,25	8

3.4 RESULTS AND DISCUSSION

The execution of smart contracts is based on the transition from one state to another after an event is triggered (BAI et al., 2018). In addition, “contracts provide functions that allow other entities (e.g., contracts or users) to invoke actions and change the status of smart contracts” (MAVRIDOU; LASZKA, 2018). For this reason, it is common to find studies that create state machine diagrams to represent how a smart contract works.

Despite this, in the search for related work, no study was found precisely related to the mapping of a formal specification language into a state diagram, although many of them defined the states of a smart contract by specifying them in a state diagram, a finite-state machine or a statechart. Furthermore, only five of them specify such diagrams according to the obligations and powers of the equivalent real-world legal contract.

Three of the selected articles focused only on the “program” published on the blockchain or related a smart contract with business models, using the Business Process Modeling Notation (BPMN) technique (AMARAL DE SOUSA; BURNAY; SNOECK, 2020). The authors in (SKOTNICA; PERGL, 2020) combined, together with BPMN, other models and concepts of enterprise engineering, such as Design and Engineering Methodology for Organizations (DEMO), and created some modified diagrams (visual language) to define a contract with regard to a possible automation of the law. In their modeling, the authors in, (SKOTNICA; PERGL, 2020) created a meta-model diagram of a contract, much like the class diagram (Unified Modeling Language (UML)), and created as well a diagram to define the process flow – a Contract Process Diagram – which allows visualizing the contract flow, respecting its axiom of transitions, as well as the state diagram, however without specifying the state of the entity.

The articles relevant to this study can be grouped according to two significance factors (Table 3): (1) those addressing the modeling of a smart contract with state

diagrams; and (2) those dealing with the legal attributes of a contract. Therefore, the related studies were divided into two subsections, as follows:

Table 3 – Subdivision of selected articles dealing with smart contract modeling or with legal contract aspects

Types	Articles	Total
Smart contract modeling (1)	1,2,4,5,7,8,9,10,12,13,16,17,18,19,20,21,22	17
Legal contract aspects (2)	3,6,11,14,15,23,24,25	8

3.4.1 Modeling with state diagram/state machine

State diagrams help in modeling any type of system, graphically defining the behavior of an object. With regard specifically to smart contracts, visualizing the behavior of a smart contract in a state diagram can facilitate its understanding, even because, as mentioned above, the creation of a smart contract can interest even people unknowledgeable about programming. State diagram modeling can also help in creating automated tests, for instance, ensuring easy visualization of what type of coverage would be needed.

This mapping of studies showed a tendency to use finite-state machines and diagrams (BPMN, PetriNets, etc.) to represent smart contracts. The creation of formal languages as a source base was also used in some of the articles, either to facilitate the understanding of a smart contract or to ensure correct modeling.

For the creation of diagrams, the main source is the contract originally written in natural language. Authors normally break down the business rules from the contract written in natural language. With regard to finding patterns in the definition of states and transitions, it was possible to observe that there is no consensus to use the same states and transitions. Figures 7 and 8, for example, show very different representations for smart contracts.

An example of a tool based on state machines is FsolidM (MAVRIDOU; LASZKA, 2018), whose authors sought to create a tool that, from the specification of a smart contract in the format of a finite-state machine, would allow automatically transcribing it into smart contract code for the Ethereum platform, written in the Solidity language. In the tool, the relationship of a smart contract with the obligations and powers of a legal contract was not explored, however, the vulnerabilities of the Ethereum platform were a concern. The application VeriSolid (MAVRIDOU et al., 2019), an improvement of FsolidM, also with a focus on generating smart contracts from fail-safe modeling (correct-by-design), derives as well from a state machine, but with the creation of a verification process of the generated models.

Two articles found bring up the issue of using software engineering techniques to model a blockchain-based system that contains smart contracts, for instance in (MARCHESI, M.; MARCHESI, L.; TONELLI, 2018), where the authors defined a step-by-step design of a blockchain-oriented application using agile methodology, defining objectives, identifying actors, and creating diagrams in the UML language. Diagrams were used, such as: class diagrams, showing the smart contract structure within the blockchain; user stories, defining roles within the blockchain; and even state diagrams (however specifying only the state of the interested parties within a contract). (GARAMVÖLGYI et al., 2018) examine the application of UML statecharts for modeling Cyber-physical systems and automatically generating smart contracts from the model. As mentioned in the introductory 1.2 section, software engineering can be considerably beneficial in modeling.

Two articles did not focus so much on the use of software engineering, but, anyway, created a formal description of smart contracts, aiming at security and attack prevention. (BAI et al., 2018) defined a formal method, extending this definition to a state machine, and used existing tools to verify the use of its formal description. This was the study that best resembled the one proposed here. Despite not seeking to automate the diagram generation, said study created a state machine, specifying the states and each transition from one state to another.

The state machine modeling proposed by (XU; FINK, 2020), besides the concern with design-by-contract where, like the concept of correct-by-design, software is designed as per contract specifications, sought mainly to avoid future flaws and vulnerabilities in the modeling of the “rights and obligations” section of a contract, using the “Temporal Logic of Actions (TLA)” security model.

3.4.2 Review of the legal aspects of the contract

Besides the fact that the search was comprehensive and encompassed any smart contract specification in the form of a state diagram, it comprised articles in which smart contract modeling included the legal aspects of a contract rather than just executing a program to check some business rules. Other four articles, in addition to mentioning legal contracts, followed a model entirely focused on them. Some of these articles are discussed below.

Despite not creating a model and aiming only at a comparison between declarative and imperative language for programming a smart contract, (GOVERNATORI et al., 2018) were incisive in stating that a smart contract can represent a legal contract, with regard to some legal aspects, such as the agreement between the parties, consideration, competence, etc. In addition, the life cycle of a legal contract was also defined, transforming it, then, into a state machine. Freedoms, obligations and powers and all legal interaction between parties to a contract were also specified

as a choreography – interactions between different participants towards a common commercial goal – with the authors in (LADLEIF; WESKE, 2019a) demonstrating relationships between interactions and legal states of a contract.

A contract modeling language (CML) was proposed by (WÖHRER; ZDUN, Uwe, 2020). Their objective was to simplify the implementation of a contract, approaching as much as possible to the natural language. They also pointed out the important common features of a legal contract that must somehow be modeled. Their result is a simple format for declaring the agreements of a contract, and it also allows generating a smart contract in the Solidity language. Their study, despite developing a domain-specific language, at no time addressed the life cycle of the contract nor its representation in a state diagram.

(WEINGAERTNER et al., 2018) proposed a new language to model contracts, also taking into account legal aspects of a contract. Similarly, (XU; FINK, 2020) also sought to model the cycle of obligations and rights within the life cycle of a smart contract.

Finally, it is worth setting forth the Symboleo contract specification language (SHARIFI, S. et al., 2020), which is used as a reference in the present study. As with some of the studies mentioned above, Symboleo is a different format that allows specifying a smart contract in an assertive way, with a language that is easier to understand, simpler than Solidity, for example. To arrive at the final notation of the language, numerous models of legal contracts of different types were studied until completeness.

3.4.3 Comparison between the main elements

Table 4 below summarizes the main aspects for comparison between the related work reviewed.

Table 4 – Comparisons between related work

	Specification of legal obligations and powers	State mapping	Automated code generation	Model verification
(BAI et al., 2018)		X		X

	Specification of legal obligations and powers	State mapping	map-	Automated code generation	Model generation	verifi-
(MARCHESI, M.; MARCHESI, L.; TONELLI, 2018)						
(GOVERNATORI et al., 2018)	X	X				
(GARAMVÖLGYI et al., 2018)		X		X		
(MADL et al., 2019)				X	X	
(XU; FINK, 2020)	X	X			X	
(BANACH, 2020)		X				
(SKOTNICA; PERGL, 2020)		X				
(MAVRIDOU et al., 2019)		X		X	X	
(MAVRIDOU; LASZKA, 2018)		X		X		
(LADLEIF; WESKE, 2019a)	X	X				
(PINNA; IBBA, 2018)		X				
(CONCHON; KORNEVA; ZAÏDI, 2020)				X	X	
(SHARIFI, S. et al., 2020)	X	X				
(WEINGAERTNER et al., 2018)		X				

	Specification of legal obligations and powers	State mapping	Automated code generation	Model verification
(ALQAHTANI et al., 2020)		X		
(ELIZONDO et al., 2019)		X		
(BOOGAARD, 2018)		X		
(HE, Xudong, n.d.)		X		
(BERTOLINI, 2020)		X		
(MITCHELL, n.d.)		X		
(CHEN, E. et al., 2021)		X	X	
(AMATO et al., 2021)	X	X		X
(FLOOD; GOODE-NOUGH, 2021)	X	X		
(JURGELAITIS; CEPONIENE; BUTKIENE, 2022)		X	X	

The aspects for such a comparison were chosen according to the aim of this study. As this study directly involves the Symboleo formal language, which is totally focused on legal contracts, it is also important that other studies that sought to specify legal obligations and powers undergo such a comparison.

State mapping is another equally important aspect, as the final product of this study is precisely the creation of a state diagram.

Automated code generation and model verification, in turn, are aspects that

appeared in a few articles, yet worth mentioning and discussing. Automation is about using a model created to automatically generate the code of a smart contract and, although it is not the focus here, it is interesting to see what modeling needs for this.

Finally, model verification refers to the fact that some of the articles used tools to test the model created. In the two articles that did so, state machines were created to represent the contract and, later, the models created were verified using methods like TLA (Temporal Logic of Actions), which “uses mathematics in a simple, native way to specify state machines; it facilitates the implementation of secure design patterns and automates vulnerability detection” (XU; FINK, 2020), or using ready-made tools, like SPIN, which is a tool used “to verify the correctness of distributed software models in a rigorous and mostly automated fashion” (BAI et al., 2018).

3.4.4 Threats to validity

With regard to possible threats to the validity of this systematic mapping, we have to mention that the systematic mapping was made by only one person, which can affect the selection of the studies. Another threat to validity is the fact that the search was divided in two different periods and it also might affect the selection of the studies.

4 PROPOSAL

The present study seeks to specify and implement an algorithm that receives as input a contract written in the domain-specific language for smart contracts called Symboleo and produces as output a state diagram corresponding to the input contract. To do so, it is necessary to find patterns in this language and map them into contract states and into transitions that connect these states.

Figure 9 shows an example of clauses of a simple contract for meat commercialization, and Figure 10 shows its formal definition using the Symboleo language.

Figure 9 – Clauses in a meat purchase and sale agreement

Meat Purchase and Sale Agreement	
Between Seller and Buyer	
This agreement is entered into as of the date <i><effDate></i> , between <i><party1></i> as Seller with the address <i><retAdd></i> , and <i><party2></i> as Buyer with the address <i><delAdd></i> .	
Terms and Conditions	
1.	Payment & Delivery
1.1	Seller shall sell an amount of <i><gnt></i> meat with <i><qlt></i> quality ("goods") to the Buyer.
1.2	Title in the Goods shall not pass on to the Buyer until payment of the amount owed has been made in full.
1.3	The Seller shall deliver the Order in one delivery within <i><delDueDateDays></i> days to the Buyer at its warehouse.
1.4	The Buyer shall pay <i><amt></i> ("amount") in <i><curr></i> ("currency") to the Seller before <i><payDueDate></i> .
1.5	In the event of late payment of the amount owed due, the Buyer shall pay interests equal to <i><intRate></i> % of the amount owed, and the Seller may suspend performance of all of its obligations under the agreement until payment of amounts due has been received in full.
2.	Assignment
2.1	The rights and obligations are not assignable by Buyer.
3.	Termination
3.1	Any delay in delivery of the goods will not entitle the Buyer to terminate the Contract unless such delay exceeds 10 Days.
4.	Confidentiality
4.1	Both Seller and Buyer must keep the contents of this contract confidential during the execution of the contract and six months after the termination of the contract.

Source: (SHARIFI, S. et al., 2020)

Figure 11 shows an example of a state diagram for the contract above, visually presenting the lifecycle of the contract. The diagram was made manually and shows how the contract behaves and the relationship between obligations and powers.

It is worth reiterating that this study intends to automatically generate state diagrams, like the one in Figure 11, from any contract written in Symboleo. Therefore, this chapter presents a proposal for mapping a contract in Symboleo into a state diagram. Initially, it will present a high-level description of an algorithm that can transform the representation of a legal contract, described in any domain-specific language for smart contracts, into a state diagram. Subsequently, the details of said algorithm for Symboleo will be presented.

To represent a state diagram, the language JSON (Javascript Object Notation) was chosen. JSON is a lightweight data-interchange format. It is easy for humans to read and write JSON models and it is easy for machines to parse and generate

Figure 10 – Meat purchase and sale agreement specified in the Symboleo language

```

Domain meatSaleD
  Seller isA Role with returnAddress: String;
  Buyer isA Role with warehouse: String;
  Currency isA Enumeration('CAD', 'USD', 'EUR');
  MeatQuality isA Enumeration('PRIME', 'AAA', 'AA', 'A');
  PerishableGood isA Asset with quantity: Number, quality: MeatQuality;
  Meat isA PerishableGood;
  Delivered isA Event with item: Meat, deliveryAddress: String, delDueD: Date;
  Paid isA Event with amount: Number, currency: Currency, from: Role, to: Role,
  payDueD: Date;
  PaidLate isA Event with amount: Number, currency: Currency, from: Role, to:
  Role;
  Disclosed isA Event with contractID : String;
endDomain

Contract meatSaleC(buyer: Buyer, seller: Seller, qnt: Number, qlt: MeatQual-
ity, amt: Number, curr: Currency, payDueDate: Date, delAdd: String, effDate: Date,
delDueDateDays: Number, intRate: Number)

  Declarations
    goods : Meat with quantity := qnt, quality := qlt;
    delivered : Delivered with item := goods, deliveryAddress := delAdd, delDueD
    := effDate + delDueDateDays;
    paid : Paid with amount := amt, currency := curr, from := buyer, to := seller,
    payDueD := payDueDate;
    paidLate : PaidLate with amount := (1 + intRate/100)×amt, currency := curr,
    from := buyer, to := seller;
    disclosed : Disclosed with contract := self;
  Preconditions
    isOwner(goods, seller);
  Postconditions
    isOwner(goods, buyer) AND NOT(isOwner(goods, seller));
  Obligations
    O1 : O(seller, buyer, true, happensBefore(delivered, delivered.delDueD));
    O2 : O(buyer, seller, true, happensBefore(paid, paid.payDueD));
    O3 : violates(O2.instance) → O(buyer, seller, true, happens(paidLate, -));
  SurvivingObls
    SO1: O(seller, buyer, true, not happens(disclosed(self), t)
    AND (t within activates(self) + 6 months));
    SO2: O(buyer, seller, true, not happens(disclosed(self), t)
    AND (t within activates(self) + 6 months));
  Powers
    P1: violates(O2.instance) → P(seller, buyer, true, suspends(O1.instance));
    P2: happensWithin(paidLate, suspension(O1.instance)) → P(buyer, seller, true,
    resumes(O1.instance));
    P3: not(happensBefore(delivered, delivered.delDueDate + 10 days)) → P(buyer,
    seller, true, terminates(self));
  Constraints
    NOT(isEqual(buyer, seller));
    forAll o | self.obligation.instance (CannotBeAssigned(o));
    forAll p | self.power.instance (CannotBeAssigned(p));
endContract

```

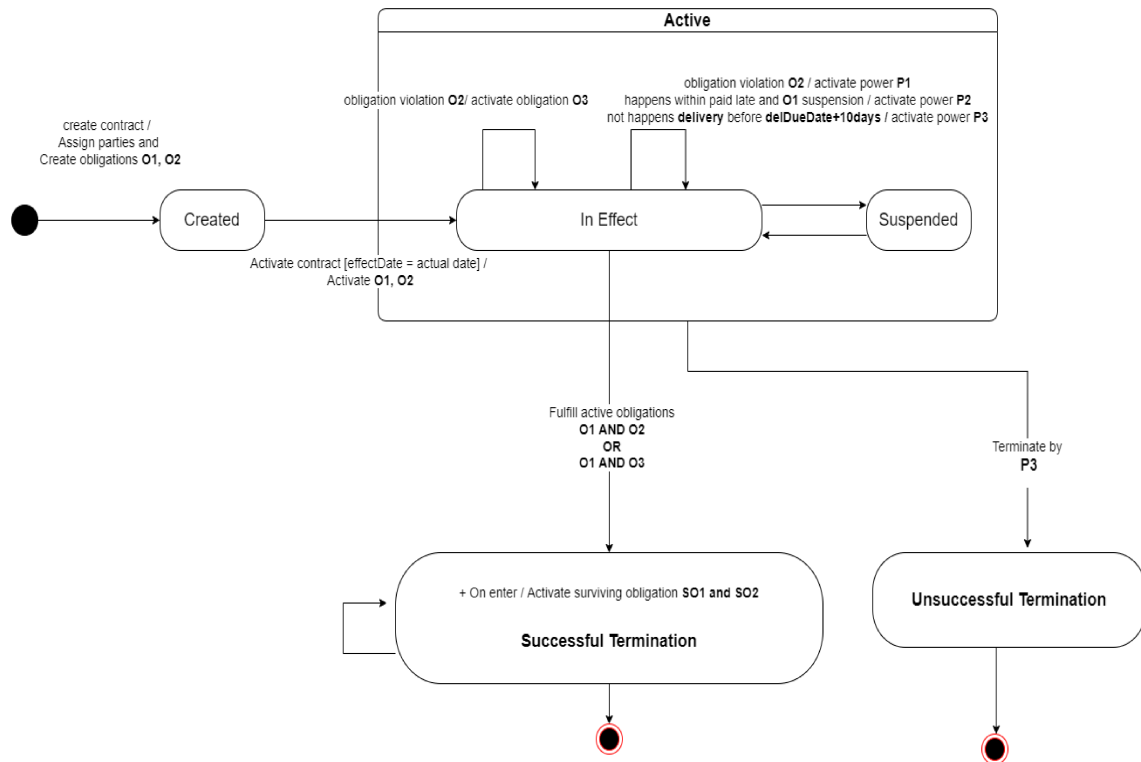
Source: (SHARIFI, S. et al., 2020)

them as well. It is also a language-independent syntax (ECMA, 2017). But any data interchange-format could be chosen, like the XMI (XML Metadata Interchange) which is an XML-based integration framework for the exchange of models, and, more generally, any kind of XML data (WEISS, 2009). Although UML is the official interchange language, JSON was selected for its ease and flexibility.

4.1 GENERAL MAPPING OF A LEGAL SMART CONTRACT INTO A STATE DIAGRAM

When it comes to creating a state diagram, it is necessary to define the states of an entity and the behavior of that entity, describing how its instances will work (AMBLER, 2003). Before presenting the steps to define a state diagram for a smart

Figure 11 – State diagram generated manually



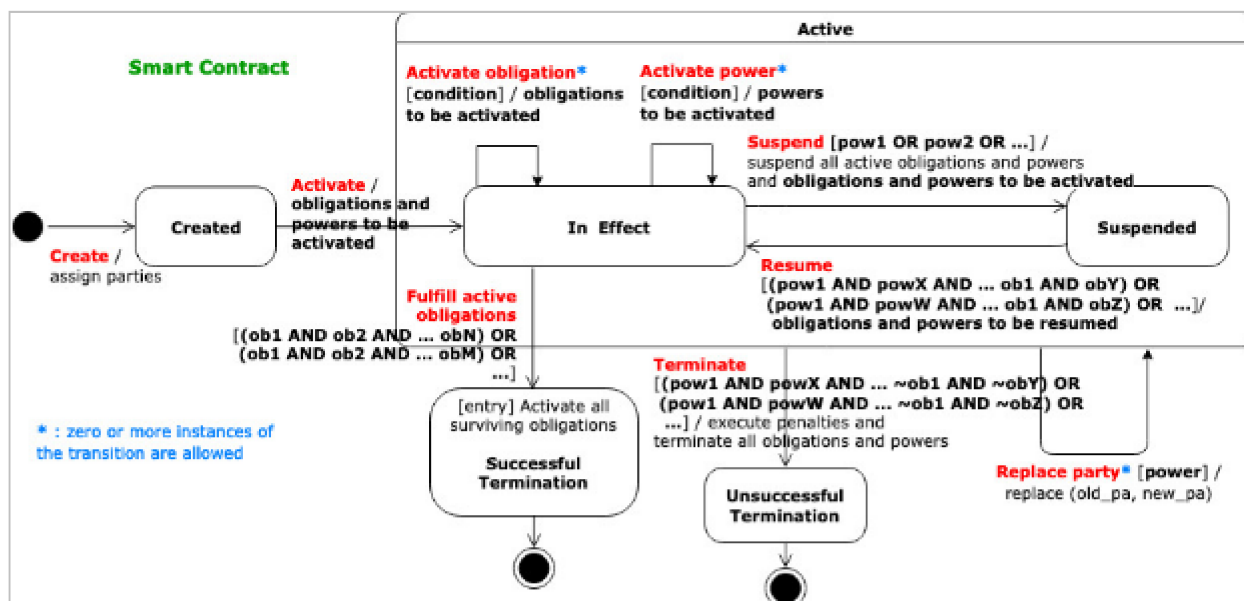
contract, the generic state diagram used as a basis in this study will be presented. The diagram also contains the requirements for a domain language to be used as input for the identification of necessary information. In addition, the algorithm that will be created also needs to understand the lexical terms, the syntax and the semantics of the domain language in order to transform the contract into a state diagram.

4.1.1 Generic state diagram for a smart contract

Figure 12 shows a generic state diagram for a smart contract. As can be seen, the states of its life cycle are already defined. In short, a smart contract, as well as a real-life contract, once it has been created, will enter into force from a given date, with the possibility of being suspended for a certain period, and will finally be successfully concluded, i.e., with due fulfillment of all clauses, or else it will be terminated unsuccessfully, when one or more clauses are “breached” for some reason, for example.

It is the terms agreed in a contract (the obligations and powers) that will allow changing its states. It can be noted that, in state diagram transitions, a given obligations or/and powers specification must be satisfied for a transition to be triggered. In other words, in the transition to activate an obligation or a power, the condition required for this clause to be activated will also have to be specified.

Figure 12 – Generic state diagram



Source: (VILAIN, 2021)

As concerns this generic state diagram, generated manually, its main objective is clearly to extract the obligations and powers of this contract, in addition to defining what is needed so that these obligations and powers can come into force and be suspended, therefore allowing the contract state to be changed.

It is important to notice that the states of the state diagram generated are always the same, for all smart contracts. It means that the complexity of a contract, written in a domain-specific language for smart contracts, does not influence on the number of states. Only the number of obligations and powers to be included in the state diagram generated can be different for different contracts.

4.1.2 Requirements of a domain language for a smart contract

According to (DIXIT et al., 2022), in order to formalize a legal contract, a domain language needs to identify normative concepts such as prohibitions, obligations, and the need for permissions.

(SHARIFI, S. S., 2020) also adds that contract languages must support various types of legal concepts, which are often based on different ontologies. Many languages use different names for the same concepts, for example, commitment, promise, obligation, and duty. Many domain-specific languages already follow existing “legal” frameworks, such as deontic logic (based on obligations, prohibitions, and permissions), UFO-L (based on Alexy’s framework), or the Hohfeld’s taxonomy of legal positions of parties.

For the creation of a state diagram of a legal smart contract from a domain-specific language, it is then necessary to:

- Define legal concepts (rights and duties);
- Define the parties involved;
- Define powers that can terminate the contract;
- Define powers that can suspend and resume the contract;
- Define obligations that must be fulfilled to successfully conclude the contract;
- Define the life cycle of each legal concept (when it is created, when it takes effect, when it is concluded or violated).

4.1.3 Steps to define a state diagram for a smart contract

Below are, in a generic way, the steps necessary to define the possible states of contract instances, and the obligations and powers associated with transitions from one state to another. All the steps presented below are independent of the domain language used as input, except for Step 4. Section 4.2 shows how Step 4 is performed based on a specification with the Symboleo language.

Step 1: Define the following states for the final state diagram: Created, In Effect, Suspended, Successful Conclusion, Unsuccessful Termination. Include transitions between states. The obligations and powers that trigger each transition will be defined later when they are extracted from the domain language.

Step 2: Describe the states and transitions, defined in Step 1, in JSON key and value format. This is important to facilitate the manipulation and replacement of the extracted data.

Step 3: Represent the data entered in JSON in an object format (contract, obligations, powers, and transition), including the necessary attributes. This step, along with the previous one, will facilitate data manipulation.

Step 4: Read the contract file written in a domain language specific to smart contracts, and extract the following information about the contract: the parties to the contract, the relevant obligations and powers, the obligations and powers to be created with the contract, the conditions for activating the powers and obligations, the powers that can end the contract, the obligations that must be fulfilled to end the contract, and the “surviving obligations” that will be activated at the end of the contract. This information must be stored in the objects that represent a contract, those created in Step 3. This step is totally dependent on the domain language used as input.

Step 5: Complete the JSON that represents the state diagram which was created in Step 2. To do so, the values of each JSON key must be replaced by the

information stored in the objects during Step 4. If there are no powers that can suspend the contract, the Suspended state, along with the transitions linked to it, must be removed from JSON.

Step 6: Finally, JSON needs to be read and transformed into the state diagram in the traditional visual way.

Figures 13, 14 and 15 with code snippet below show the generic JSON defined in Step 2 and used as the basis for the algorithm. With the JSON defined, the main goal of the algorithm is to complement some attributes of the transitions, mainly those naming which obligations and powers are responsible for triggering such transitions, thus changing the contract state. This is important because each transition is directly linked to the execution of powers or fulfillment of obligations.

Thinking of creating a generic model and facilitating the subsequent step of transforming a formal contract into a diagram in the visual format, all transitions have the “events” attribute (which is a set of events), formed by an event that triggers the transition; the “guard” attribute, which is when there is a condition for the event to occur; and the “actions” attribute, which are actions that will be performed along with the transition, that is, consequences for the event in question. Actions and events may have attributes in their description which will need to be replaced by other values. These attributes are always inserted with the following syntax: *\$attribute_name*, where the name of this attribute is also a key within the transition and the values inserted in this key in the execution of the algorithm will have the content that will compose the actions/events.

Each attribute that must be filled in by the algorithm is explained below:

- The “create_contract” transition has three actions: designate the parties and include the obligations and powers that will be created along with the creation of the contract, so it is necessary to complement the transition with these parties, obligations and powers;
- The “activate_contract” transition needs to include the obligations and powers that will be activated along with the contract activation;
- The “activate_obligation_power” is a transition between the same state (In Effect) and needs to be filled in with all events that create a conditional obligation or power that might be triggered during the life cycle of the contract. The “event” attribute itself is generic because it must specify whether the event will be an obligation that was violated, a power that must be exercised or another condition that needs to be satisfied for the activation of an obligation or power;
- The “suspend_contract” transition needs to be complemented with information regarding powers that suspend the contract, and obligations and

Figure 13 – Json Model

```
1  {
2    "name": "${name}",
3    "states": {
4      "created": "Created",
5      "in_effect": "In Effect",
6      "suspended": "Suspended",
7      "successful_termination": "Successful Termination",
8      "unsuccessful_termination": "Unsuccessful Termination"
9    },
10   "transitions": {
11     "create_contract": {
12       "source": "initial",
13       "target": "created",
14       "events": [{
15         "event": "Create",
16         "guard": "",
17         "actions": [
18           "Assign parties ${parties}",
19           "Create obligations ${obligations}",
20           "Create powers ${powers}"
21         ]
22       }],
23       "parties": [],
24       "obligations": [],
25       "powers": []
26     },
27     "activate_contract": {
28       "source": "created",
29       "target": "in_effect",
30       "events": [{
31         "event": "Activate",
32         "guard": "effectiveDate = actualDate",
33         "actions": [
34           "Activate obligations ${obligations}",
35           "Activate powers ${powers}"
36         ]
37       }],
38       "obligations": [],
39       "powers": []
40     },
```

Figure 14 – Json Model Transition

```
1  "fulfill_active_obligations": {
2      "source": "in_effect",
3      "target": "successful_termination",
4      "events": [{
5          "event": "Fulfill active obligations ${set_of_obligations}",
6          "guard": "",
7          "actions": []
8      }],
9      "set_of_obligations": [""],
10 }
```

Figure 15 – Json Model Actions

```
1  "state_actions": {
2      "successful_termination": {
3          "when": "+ On Enter",
4          "action": "Activate surviving obligations ${surviving_obligations}",
5          "surviving_obligations": [""],
6      },
7      "unsuccessful_termination": {
8          "when": "+ On Enter",
9          "action": "Activate surviving obligations ${surviving_obligations}",
10         "surviving_obligations": [""],
11     }
12 }
```

powers that will be activated upon contract suspension;

- The "resume_contract" transition needs as well to define which powers can resume the contract to the active state, in addition to the obligations and powers that will be activated upon contract resumption;
- The "replace_party" transition needs to define which powers can replace any of the parties of the contract, assigning a new party to the contract;
- The "fulfill_active_obligations" transition needs to be complemented with the set of obligations that must be fulfilled for the contract to be successfully concluded. In this case, it is necessary to find out the possible sets (which may be more than one) of obligations that are active and that must be fulfilled together.
- The "activate_surviving_obligation" transition needs to be complemented with the surviving obligations (obligations that last even after the end of the contract, for example, non-disclosure agreements) that depend on some event to be active upon successful conclusion of the contract;
- The "terminate_contract" transition needs to include the powers that can terminate the contract and which obligations/penalties must be activated when terminating the contract.

The JSON also has the "state_actions" key, which defines actions that will be performed when entering a certain state. In the case of the JSON model in question, actions can be performed in the states "successful_termination" and "unsuccessful_termination" and these actions are related to the activation of surviving obligations, that is, as mentioned above, obligations that must exist and be fulfilled either upon contract termination or successful conclusion.

4.2 MAPPING A SYMBOLEO SMART CONTRACT INTO A STATE DIAGRAM

The steps mentioned in the previous section must be adapted to the domain language used to represent a smart contract. This is because each language has its specification of how to represent relevant data and, therefore, the information needed to define the state diagram can be represented in different ways in each language. As the present study uses the Symboleo language, this language will be introduced subsequently. After that, there will be a description of how the mapping was done from a representation with the Symboleo language.

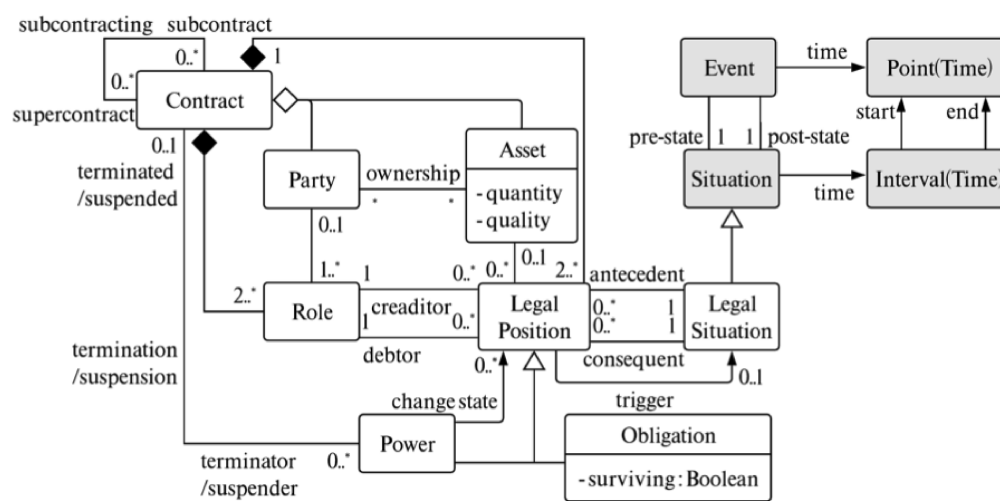
4.2.1 The Symboleo language

The Symboleo language was designed through an in-depth study into legal elements of contracts (SHARIFI, S. et al., 2020). And as mentioned in (SHARIFI, S. S.,

2020): "A contract is a promise or a set of promises that are legally enforceable and, if violated, allow the injured party access to legal remedies. Contract law recognizes and governs the rights and duties arising from agreements".

Based on that study, a legal ontology for Symboleo was defined, as shown in Figure 16. With this ontology, it was possible to specify and adapt Step 4 to the Symboleo language. As we can observe below, the Symboleo language is able to implement the legal aspects, such as obligations and powers, in a way that these obligations and powers can be interpreted by computers.

Figure 16 – The proposed Symboleo language ontology



Source: (SHARIFI, S. et al., 2020)

The first block of the language is called Domain and specifies entities that will be inserted into the contract and their attributes. These entities are specializations of the ontology concepts mentioned above. Then, a domain object is created with a list of entities. For each entity, a name and a specialization are defined, as well as a list of attributes that represent that entity along with the domain of each attribute. For example, in Figure 17, the first entity in the domain is "Seller", its specialization is "Role", and it contains the "returnAddress" attribute.

After the Domain block, the contract is created, along with all the parameters that will need to be passed in order to create said contract. At first, it is necessary to identify the parties to the contract, that is, parameters of the "Role" type, or a specialization of "Role". In the example shown in Figure 18, the parties appear in a "Buyer" and "Seller" format, specified in Domain.

Subsequently, it is necessary to specify the obligations, powers and "surviving obligations". These three terms, also called "legal situations" by the authors of Symboleo, will probably have the most relevant information for our state diagram, as it is

Figure 17 – Section of the Domain block specified in Symboleo

```

Domain meatSaleD

  Seller isA Role with returnAddress: String;
  Buyer isA Role with warehouse: String;
  Currency isA Enumeration('CAD', 'USD', 'EUR');
  MeatQuality isA Enumeration('PRIME', 'AAA', 'AA', 'A');
  PerishableGood isA Asset with quantity: Number, quality: MeatQuality;
  Meat isA PerishableGood;
  Delivered isA Event with item: Meat, deliveryAddress: String, delDueD: Date;
  Paid isA Event with amount: Number, currency: Currency, from: Role, to: Role,
  payDueD: Date;
  PaidLate isA Event with amount: Number, currency: Currency, from: Role, to:
  Role;
  Disclosed isA Event with contractID : String;

endDomain

```

Source: (SHARIFI, S. et al., 2020)

Figure 18 – Contract parametrization in Symboleo

```

Contract meatSaleC(buyer: Buyer, seller: Seller, qnt: Number, qlt: MeatQual-
ity, amt: Number, curr: Currency, payDueDate: Date, delAdd: String, effDate: Date,
delDueDateDays: Number, intRate: Number)

```

Source: (SHARIFI, S. et al., 2020)

these objects that influence the life cycle of a contract.

Figure 19 shows the blocks in which obligations, surviving obligations and powers are described.

As can be seen, in the section where obligations are defined (Obligations), every obligation in Symboleo is written in the $O_i d:O(\text{debtor}, \text{creditor}, \text{antecedent}, \text{consequent})$ format. Debtor and creditor are roles within a contract and means that the obligation is the responsibility of the debtor to the creditor. Antecedent and consequent are legal situations (a situation associated with the contract), in other words, antecedent and consequent propositions describe situations that need to be satisfied in order for obligations to be fulfilled (SHARIFI, S. et al., 2020).

4.2.2 Detail of the mapping from a Symboleo specification

The following details how a Symboleo specification can be mapped into a state diagram. As mentioned earlier, only Step 4 of the mapping needs to be adapted for each smart contract domain language. So, only this step will be detailed below.

Figure 19 – Legal situations defined in Symboleo

```

Obligations
O1 : O(seller, buyer, true, happensBefore(delivered, delivered.delDueD));
O2 : O(buyer, seller, true, happensBefore(paid, paid.payDueD));
O3 : violates(O2.instance) → O(buyer, seller, true, happens(paidLate, -));
SurvivingObls
SO1: O(seller, buyer, true, not happens(disclosed(self), t)
AND (t within activates(self) + 6 months));
SO2: O(buyer, seller, true, not happens(disclosed(self), t)
AND (t within activates(self) + 6 months));
Powers
P1: violates(O2.instance) → P(seller, buyer, true, suspends(O1.instance));
P2: happensWithin(paidLate, suspension(O1.instance)) → P(buyer, seller, true,
resumes(O1.instance));
P3: not(happensBefore(delivered, delivered.delDueDate + 10 days)) → P(buyer,
seller, true, terminates(self));

```

Source: (SHARIFI, S. et al., 2020)

Step 4 describes the following: *Read the contract file written in a domain language specific for smart contracts and extract the following information about the contract: the contract parties, the obligations and powers thereof, the obligations and powers to be created with the contract, the conditions for activation of the powers and obligations, the powers that can end the contract, the obligations that must be fulfilled to end the contract, and the “surviving obligations” that will be activated at the end of the contract. This information must be stored in the objects that represent a contract, which were created in Step 3.*

The challenge in this endeavor is to map the states and know what needs to be prompted to cause an exchange of states, that is to say, it is necessary to understand the relationship between entities and translate the legal situations. For example, in Figure 19, obligation O₃ will be created when what is written to the left of the arrow happens, but what actually happens for obligation O₂ to be violated? After trying to understand the logic and ontology of the language, it is already known that such a legal situation means the transition of obligation O₂ from the “In Effect” state to the “Violation” state and it is also known, according to the axioms defined by the authors of the language, that such a violation occurs when its consequent has a term and such a term has expired.

The following shows how each required information is identified from a Symboleo specification.

Domain: as already mentioned, it is necessary to understand the domains that will be included in the contract, so there is the Domain entity that has the attributes:

- Name: the domain name;

- Specialization: the specialization of this domain in relation to the ontology already mentioned;
- Attributes: the attributes of this domain.

At first, by reading the domain, the purpose is to understand which contract domains are specializations of the “Role” entity, and only then specify in the diagram the parties involved in said contract.

Legal positions: obligations and powers are treated, in the ontology mentioned above, as legal situations. They represent the duties and rights of one party in relation to another. In the Symboleo language, they are represented in the same way, therefore there is an object called Legal Position which has the following attributes:

- Name: the name to identify the obligation or power. For example, O1 or P1, as displayed in Figure 19.
- Debtor: represents one of the parties to the contract. In an obligation, it determines the party that is responsible for performing the obligation, and in a power, it determines the party that will be affected by that power.
- Creditor: also represents one of the parties to the contract. In an obligation, it determines the party that will benefit from the obligation, and in a power, it determines who has the right to exercise the power.
- Antecedent: a proposition that describes a situation that must occur for an obligation or power to take effect.
- Consequent: it is also a proposition that describes a situation which, when true, means that the obligation has been fulfilled or, in the case of a power, means that it has been exercised.
- Trigger: it is a non-mandatory attribute that also describes a situation which must happen for an obligation or power to take effect. As such, it may be that the legal position in question does not need to exist throughout the life cycle of the contract.

Obligations: an obligation is a specialization of a Legal Position that has all the attributes described above and also a signal to specify whether or not this obligation is a “surviving obligation”, i.e., an obligation that may continue in force even after conclusion of the contract. Then, a new instance of the Obligation object is created for each obligation inside Symboleo’s Obligations block, identifying and defining each of its attributes.

For example, obligation O₃ is the buyer’s obligation to the seller for making late payments. Unlike O₁ and O₂, it has a proposition at the beginning of its statement, which is a condition to be created. The way of describing propositions in itself shows a relationship of order and effect. It is necessary that whatever is ahead of the arrow be carried out in order for the obligation to be created.

Powers: power is also a Legal Position and has all the attributes mentioned above and none more.

Figure 19 shows the three powers of the contract. These powers have all the attributes mentioned in the legal positions. For example, the seller's power (right) P_1 , vis-à-vis the buyer, to suspend their obligation to deliver the goods. But this seller's power will only take effect if the buyer has not fulfilled their obligation to pay on the scheduled date, that is, when a violation of O_2 takes place.

Contract: the contract entity was created to facilitate diagram making, and it groups the attributes necessary for the creation of said diagrams:

- Name: the name given to the contract in the block in which it was stated, as shown in Figure 18.
- Parties: a list with the names of the parties involved in the contract.
- Obligations: the list of contract obligations, which are the legal situations extracted as obligations.
- Powers: the list of contract powers, which are the legal situations extracted as powers.

Transition: this entity was created only to help create the state diagram and determine the transitions between states. It has the following attributes common to a transition event within a state diagram:

- Event: the event that needs to happen for the transition between states to occur.
- Guard: a condition that can exist for the event to occur.
- Actions: a list of actions that are triggered as soon as the event occurs.

4.3 MAPPING IMPLEMENTATION

Regarding the mapping implementation, two parts are necessary: the first is the creation of an API (Application Programming Interface), which is written in Node JS, with a single resource that receives as input a file of text written in Symboleo. This feature will invoke an algorithm whose processing involves reading the parts of the text through regular expressions. This reading needs to capture both the attributes of the entities and the situations that must become transitions in the diagram. The generated output must be an object written with the language JSON (JavaScript Object Notation) or YAML (YAML Ain't Markup Language), so that, further on, its result can be easily read in any other language by any program and can be turned into a state diagram in a visual format. In the second part, to complement the development thereof, an interface (Frontend) is also created, one that reads the return JSON and visually transforms it into a state diagram. This interface will be a web application with the React JavaScript library.

4.3.1 API Project

Figure 20 and Figure 21 show, respectively, a first version of the class diagram and the sequence diagram of the aforementioned API implementation, including the main classes. The class that receives the request with the contract file upload is called ContractController. This controller invokes two main classes that work as services, that is, it holds the business rules of the transformation. The ContractTransformService class is responsible for reading the file and populating the contract attributes and subclasses. With the contract object properly created, the ContractDiagramService class is able to transform the objects by reading their attributes and behavior, into a state diagram or, in the case of the API in question, into a JSON object that represents this diagram.

Figure 20 – Class diagram API

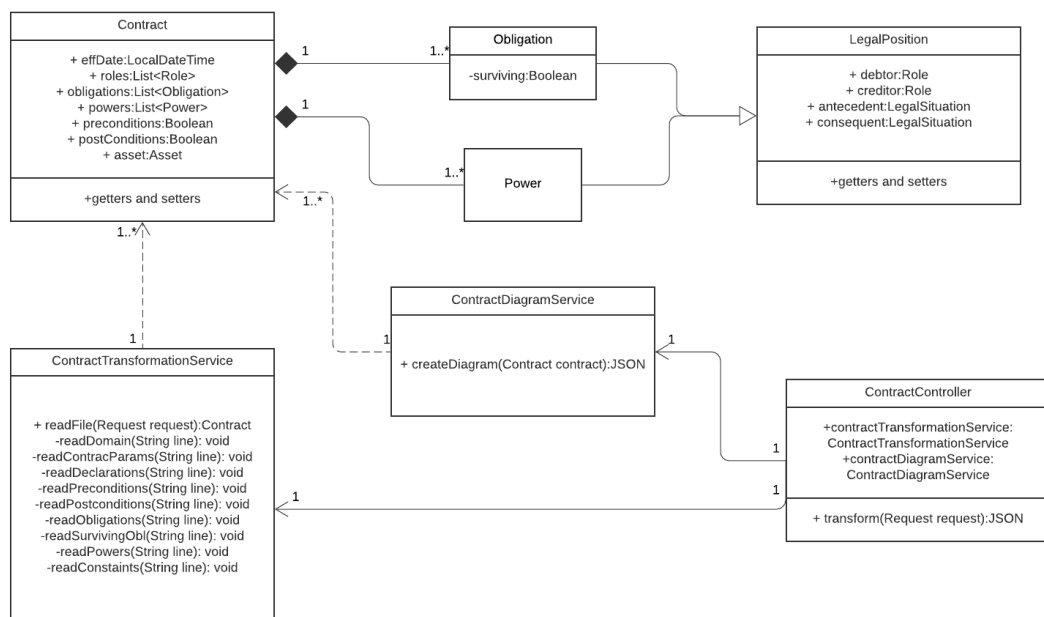
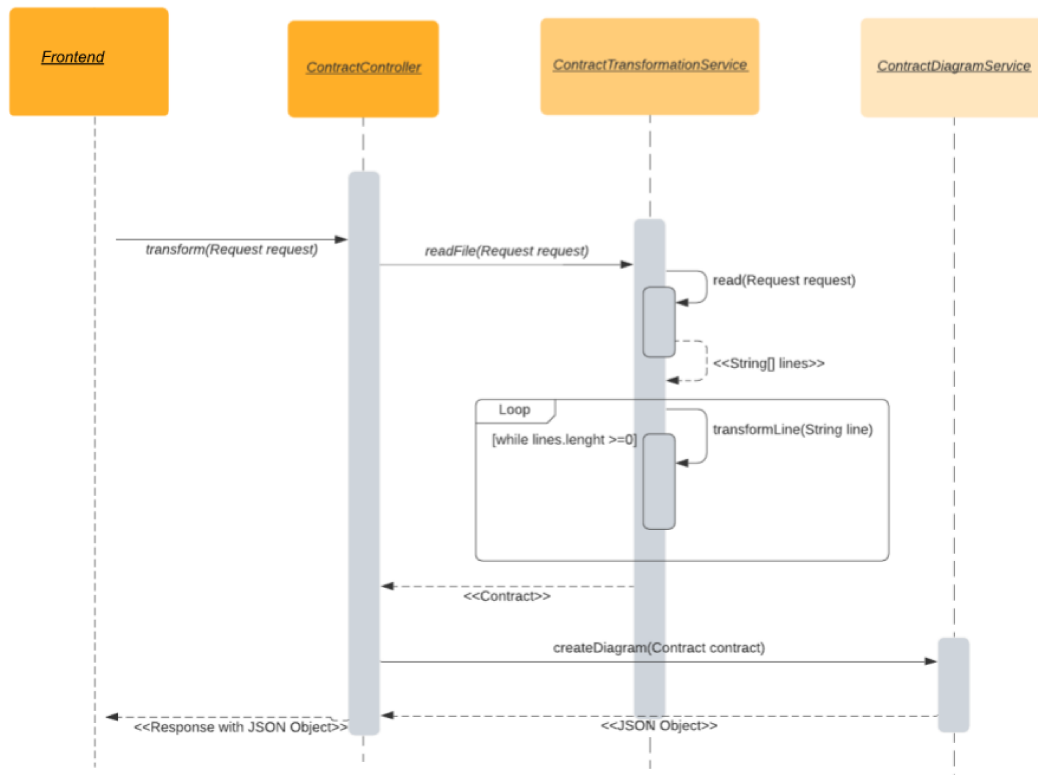


Figure 21 displays in more detail the flow that the algorithm must follow and shows the list of the most relevant methods and their respective returns.

4.3.2 API implementation

When a file is received with the contract written in Symboleo, it is necessary to “read” and break down each of the powers and obligations (called legal positions), which, by definition, always specify (1) the parties involved (the power/obligation of a party towards another), (2) a condition for this legal position to come into effect

Figure 21 – Sequence diagram API



(antecedent), and (3) a condition that specifies what needs to happen for it to be fulfilled/exercised.

As regards its implementation, the API model is proceeded, using the language Node JS (JavaScript) with Typescript (typing in JavaScript), as previously mentioned. Some software design patterns were implemented, as follows: for the creation of classes, factories were implemented; and for the behavior of the entities, commands were used.

For now, the API of the proposed mapping consists of only one feature: transforming a domain language specific for a smart contract into a state diagram. Nevertheless, to achieve this goal, there are two major steps to follow, with some independent business rules: (1) reading the smart contract file written with a domain language and capturing the main entities (obligations and powers); and (2) transforming the behavior of these entities into a state diagram.

Figures 22 and 23 show the diagrams of the implemented API. In the methods or attributes where the “Object” type appears, it is an attribute of the JavaScript object type, that is, an object formed by a certain amount of key and value, and these keys and values will only be known when a few lines from the input file are read.

With the class diagram and sequence diagram ready, together with the man-

Figure 22 – Class Diagram API

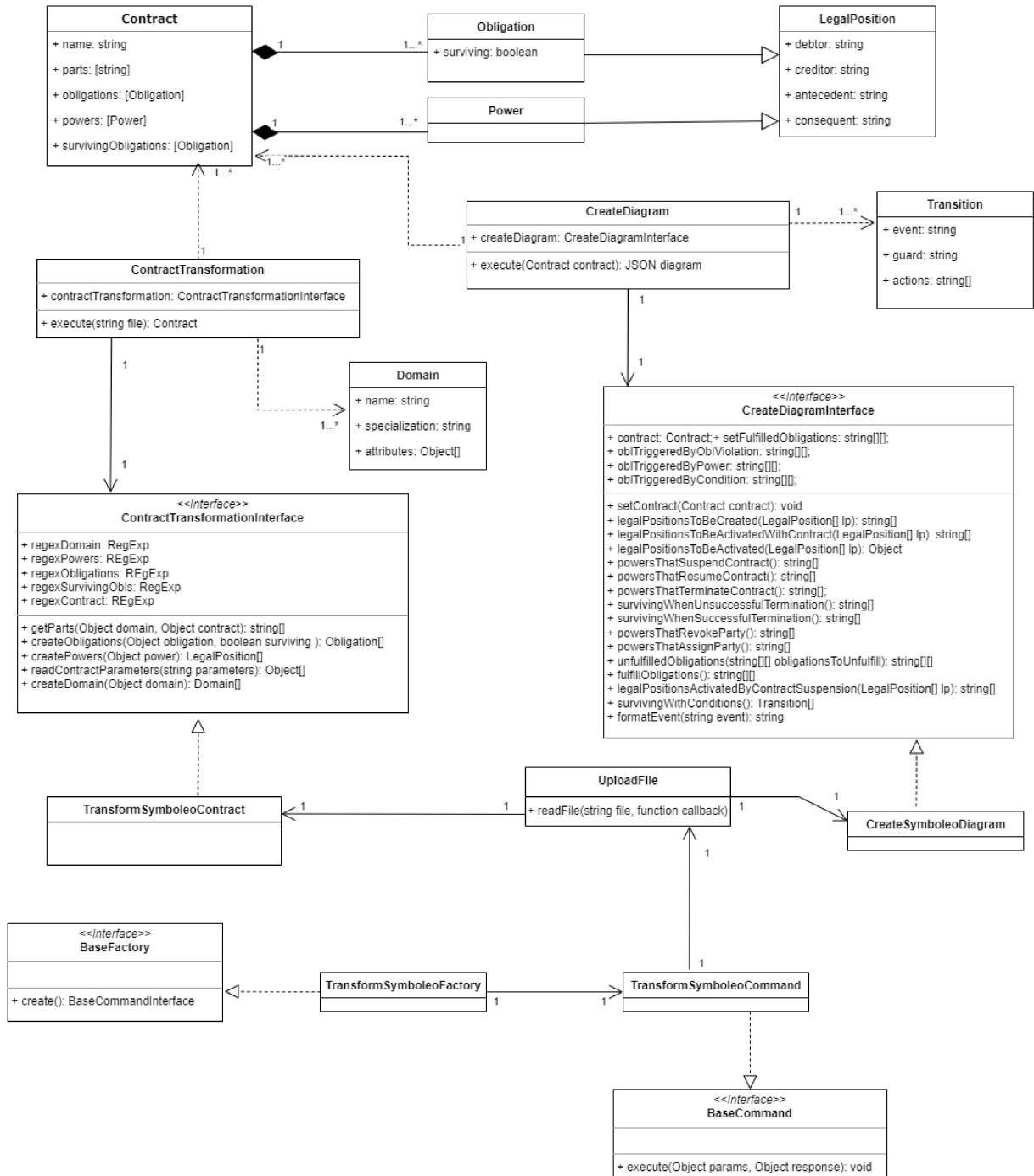
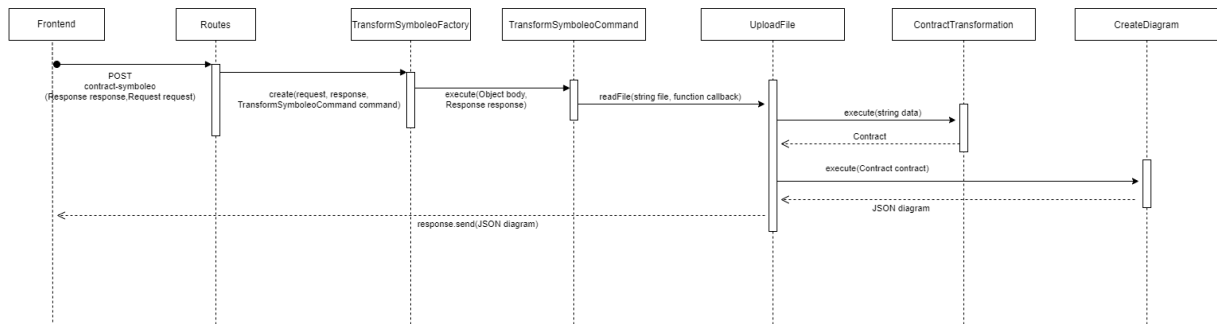


Figure 23 – Sequence Diagram API



ually generated state diagram (Figure 11), it is now possible to map the pre-defined transitions and fill in some keys from the JSON file that represents the state diagram with their corresponding values. The final JSON generated from the specification of the contract mentioned above can be seen in Figures 24, 25 and 26 below.

As can be seen in Figure 24, in lines 20, 21 and 22, there are transitions and each one of them has attributes that must be replaced. What the algorithm does is include in the keys of this transition the values that should replace that specific token, i.e., where there is token $\${parties}$ (line 20), there will be a replacement by the values inside the key with the same name “parties” (line 26).

In Figure 25, in the “activate_obligation_power” transition, the events are already completely filled in, because the event itself and the consequent action of the event are discovered only at runtime, as previously indicated, and it makes no sense to specify values where it will only be necessary to replace certain attributes. It could be the violation of a specific obligation, or simply any other event, as in line 34, for example. Some words are between the “strong” tag (bold style) and it is made just to highlight the words that represent the events on the frontend.

The biggest challenge of the algorithm is to interconnect obligations and powers with their corresponding dependencies (other obligations and powers). For example, for a contract to be enforced, it is necessary to know which set of obligations must be fulfilled, and there may be more than one possibility. An obligation “o2” can only be created if another obligation “o1” has not been fulfilled, hence it is already known that there will be two sets of possibilities in the execution of this contract: (i) a set of obligations to be fulfilled including obligation “o1” and (ii) another set of obligations where obligation “o2” replaces obligation “o1” when the latter is not fulfilled. In this case, both obligations cannot co-exist in this scenario of obligations to be fulfilled during the contract lifecycle: (*fulfill(O1) || NOTfulfill(O1) and fulfill(O2)*).

The above explanation demonstrates what is done to fill the *fulfill_active_obligations* transition in presented in the code snippet with the final JSON response above. The algorithm identified that obligations named O1 and O2 are the ones that are created

Figure 24 – Final JSON

```
1  {
2    "name": "meatSaleC",
3    "states": {
4      "created": "Created",
5      "in_effect": "In Effect",
6      "suspended": "Suspended",
7      "successful_termination": "Successful Termination",
8      "unsuccessful_termination": "Unsuccessful Termination"
9    },
10   "transitions": {
11     "create_contract": {
12       "source": "initial",
13       "target": "created",
14       "events": [
15         {
16           "event": "Create",
17           "guard": "",
18           "actions": [
19             "Assign parties ${parties}",
20             "Create obligations ${obligations}",
21             "Create powers ${powers}"
22           ]
23         }
24       ],
25       "parties": [
26         "buyer",
27         "seller"
28       ],
29       "obligations": [
30         "01",
31         "02"
32       ],
33       "powers": []
34     },
35     "activate_contract": {
36
```

Figure 25 – Final JSON transition



```

1  "activate_obligation_power": {
2    "source": "in_effect",
3    "target": "in_effect",
4    "events": [
5      {
6        "event": "Obligation violation <strong>payment</strong>",
7        "guard": "",
8        "actions": [
9          "Activate obligation <strong>latePayment</strong>"
10       ]
11     },
12     {
13       "event": "Obligation violation <strong>payment</strong>",
14       "guard": "",
15       "actions": [
16         "Activate power <strong>suspendDelivery</strong>"
17       ]
18     },
19     {
20       "event": "Obligation violation <strong>delivery</strong>",
21       "guard": "",
22       "actions": [
23         "Activate power <strong>terminateContract</strong>"
24       ]
25     },
26     {
27       "event": "Obligation violation <strong>delivery</strong>",
28       "guard": "",
29       "actions": [
30         "Activate power <strong>resumeContract</strong>"
31       ]
32     },
33     {
34       "event": "Happens <strong>paidLate</strong> Within <strong>Suspension of delivery</strong>",
35       "guard": "",
36       "actions": [
37         "Activate power <strong>resumeDelivery</strong>"
38       ]
39     }
40   ]
41 },

```

with the contract and need to be fulfilled. The algorithm also identified that obligation O2 is replaced by obligation O3 if O2 is violated somehow. This can be confirmed by referring to the definition of obligations within the contract written in Symboleo.

Understanding each type of obligation is the main role of the algorithm in order to define the possible sets of obligations that must be fulfilled for the successful completion of a contract. First, it is necessary to identify whether or not an obligation has a trigger to be created; if not, it is already in the set of active obligations that are created along with the contract. Then, when the obligation has this trigger, it is necessary to identify the meaning of this trigger: Is it the violation of another obligation? Is it the exercise of a power? Symboleo allows identifying such events because the language specifies some tokens to represent events related to obligations, powers

Figure 26 – Final JSON actions

```

1  "state_actions": {
2    "successful_termination": {
3      "when": "+ On Enter",
4      "action": "Activate surviving obligations ${surviving_obligations}",
5      "surviving_obligations": [
6        "S01",
7        "S02"
8      ]
9    },

```

and the contract itself, e.g., *oVIOLATION(O1)*, which means the violation of the obligation named O1. The pseudocode of the *fulfilledObligations* algorithm is described in the code snippet of algorithm 1 below:

Algorithm 1 Fulfill active obligations

```

1: function fulfilledObligations
2:   setOfFulFilledObligations ← new multidimensional array
3:   arrayOfActiveObligations ← findObligationsToBeCreated()
4:   obligationsWithTrigger ← findObligationsWithTrigger()
5:   replacementObligationsMap ← getReplacementMap(obligationsWithTrigger)
6:   obligationsTriggeredByCondition ← new array
7:   for obligationinobligationsWithTrigger do
8:     if replacementObligationsMap not contains obligation.name as key then
9:       obligationsTriggeredByCondition add obligation
10:    end if
11:  end for
12:  setOfFulFilledObligations add arrayOfActiveObligations
13:  setOfFulFilledObligations add obligationsTriggeredByCondition
14:  replaceFulFilledObligations(replacementObligationsMap,
    obligationsWithTrigger, setOfFulFilledObligations)
15: end function

```

In brief, the algorithm needs to complement the necessary information mentioned above, so there is a skeleton with the functions that need to be implemented, both with regard to the breakdown of the contract and the assembly of the diagram in JSON format. For now, these functions are implemented for the Symboleo language; however, our aim is to create an algorithm able to facilitate the extension and main-

Algorithm 2 Replace fulfilled obligations

```

function replaceFulfilledObligations(replacementObligationsMap,
obligationsWithTrigger, setOfFulfilledObligations)
2:  obligations ← all obligations of contract
   newSets ← new multidimensional array
4:  notFoundedObligations ← new list of obligation
   for obligation in obligations do
6:      if replacementObligationsMap contains obligation.name as key then
           founded ← false
8:      obligationToReplace ← replacementObligationsMap.get(obligation.name)
           for fulfilledObligations in setFulfilledObligations do
10:         obligationIndex ← index of obligationToReplace in fulfilledObligations
           if obligationIndex > 0 then
12:             founded ← true
               newSet ← copy of fulfilledObligations
14:             newSet[obligationIndex] ← obligation.name
               newSets add newSet
16:
           end if
18:       end for
           if founded is false then
20:             notFoundedObligations add obligation
           end if
22:       setOfFulfilledObligations add all newSets
           end if
24:     end for
           if notFoundedObligations has elements then
26:         replaceFulfilledObligations(replacementObligationsMap,
           notFoundedObligations, setFulfilledObligations) //recursive call in case obligation
           is not in list yet
           end if
28: end function

```

tenance for other formal languages. The complete implementation of the algorithm is available in the gitHub repository ¹.

4.3.3 Frontend

To facilitate the use of the algorithm and submit any contract written in Symboleo (with .txt extension), a simplified web application was created with the sole objective (for now) of submitting a file and automatically obtaining the visual state diagram.

Figures 27, 28 and 29 show the interface created to select a file from a contract written in Symboleo. As another domain-specific language still cannot be used, it is possible to observe that the Symboleo option is already selected.

¹ <https://github.com/marina luiza/contract-api>

Figure 27 – Frontend Home

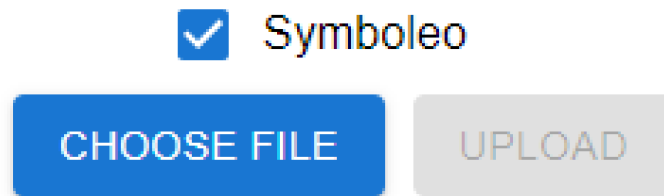


Figure 28 – Select file in Frontend

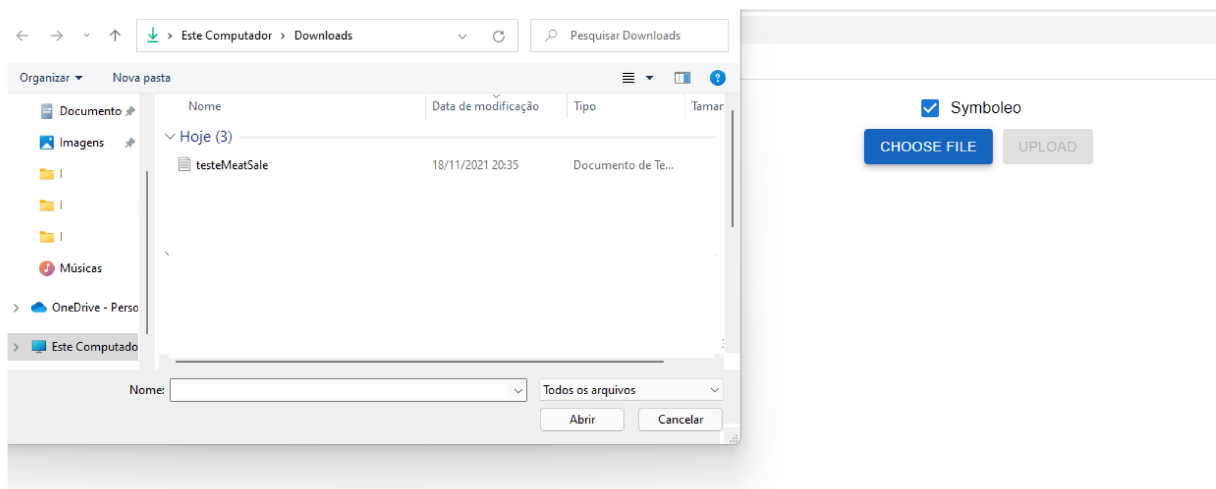


Figure 29 – File selected in Frontend

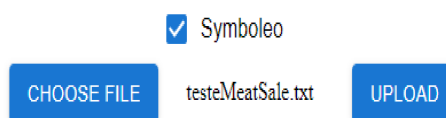


Figure 30, then, shows the automatically generated state diagram after the file is properly loaded. To generate the state diagram, an auxiliary library called Mermaid² was used, which allows generating visual diagrams through text and code. Figure 31 illustrates, in a very simplified way, the text format expected by Mermaid to generate a state diagram. The documentation shows other resources available in the library. The frontend project has a helper method that transforms the diagram in JSON format into the format expected by the Mermaid library.

The complete implementation of the frontend is available in the gitHub repository³.

Figure 30 – State diagram generated in Frontend

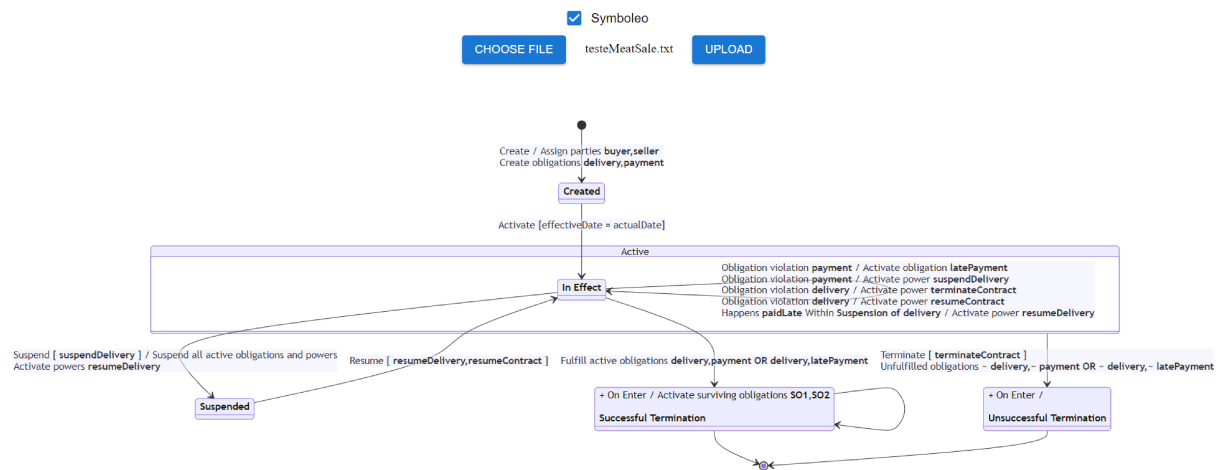


Figure 31 – Text for state diagram creation with the Mermaid library

```
stateDiagram-v2
    [*] --> Still
    Still --> [*]

    Still --> Moving
    Moving --> Still
    Moving --> Crash
    Crash --> [*]
```

² <https://mermaid-js.github.io/mermaid/#/>

³ <https://github.com/marinaluiza/contract-frontend>

4.3.4 Development evaluation

To evaluate the correct generation of the contract, we can do a comparison between the diagrams generated manually and the diagrams generated automatically using the API. Figure 32 and 33 shows the meat sale diagrams mentioned before, the diagram generated manually is on 32 and the state diagram generated automatically is on 33. Besides the differences in style and organization, it is possible to observe that the states and transitions are pretty similar, i.e., the transitions are all the same and they link the same sources and targets.

Figure 34 and 35 shows another comparison of a different contract. Again, besides the placement of components is not well organized as when the diagram is manually generated, the states and transitions are correct and reflect the right diagram lifecycle.

Figure 32 – Meat sale manually generated state diagram

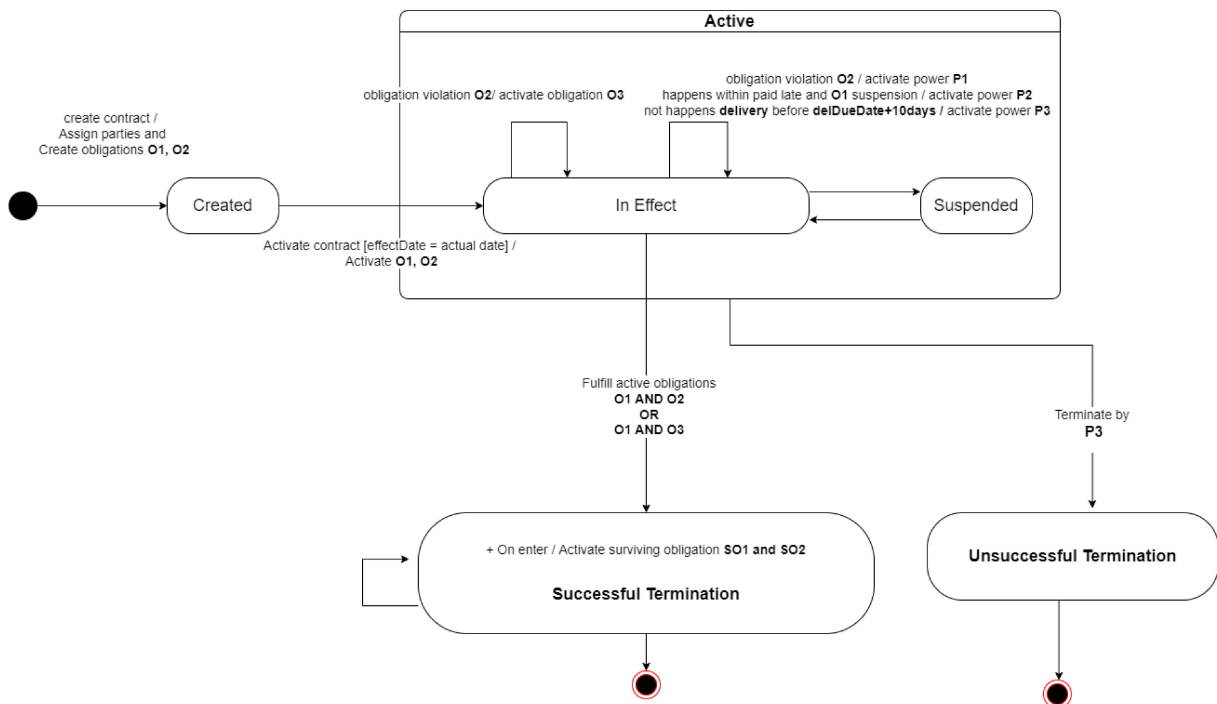


Figure 33 – Meat sale automatically generated state diagram

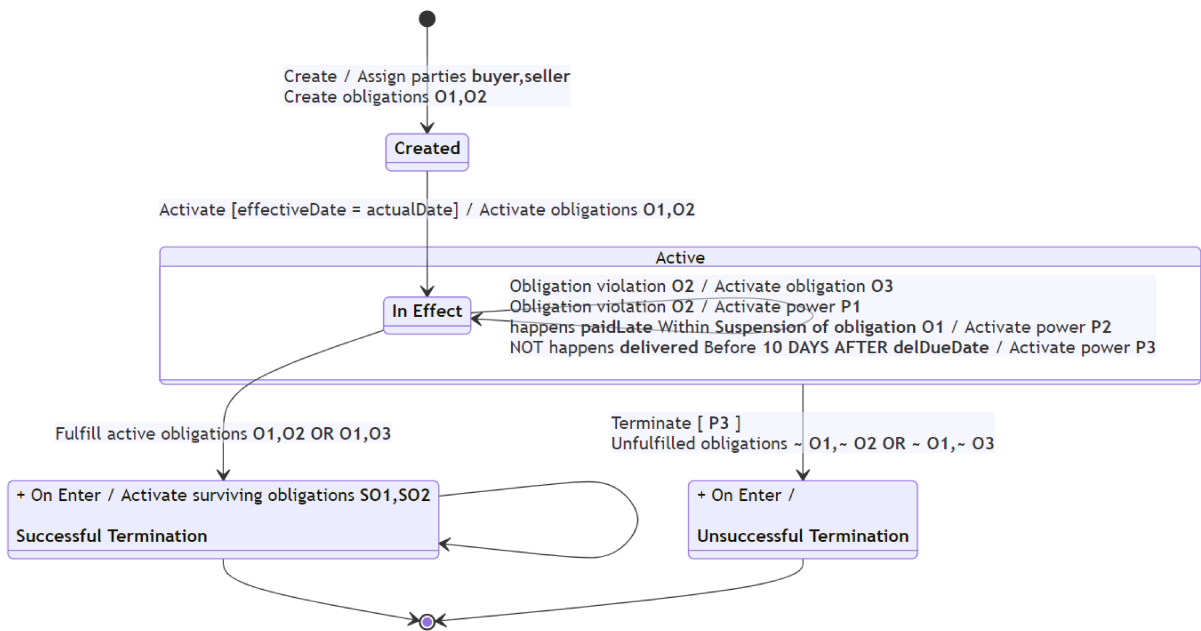


Figure 34 – Tech service manually generated state diagram

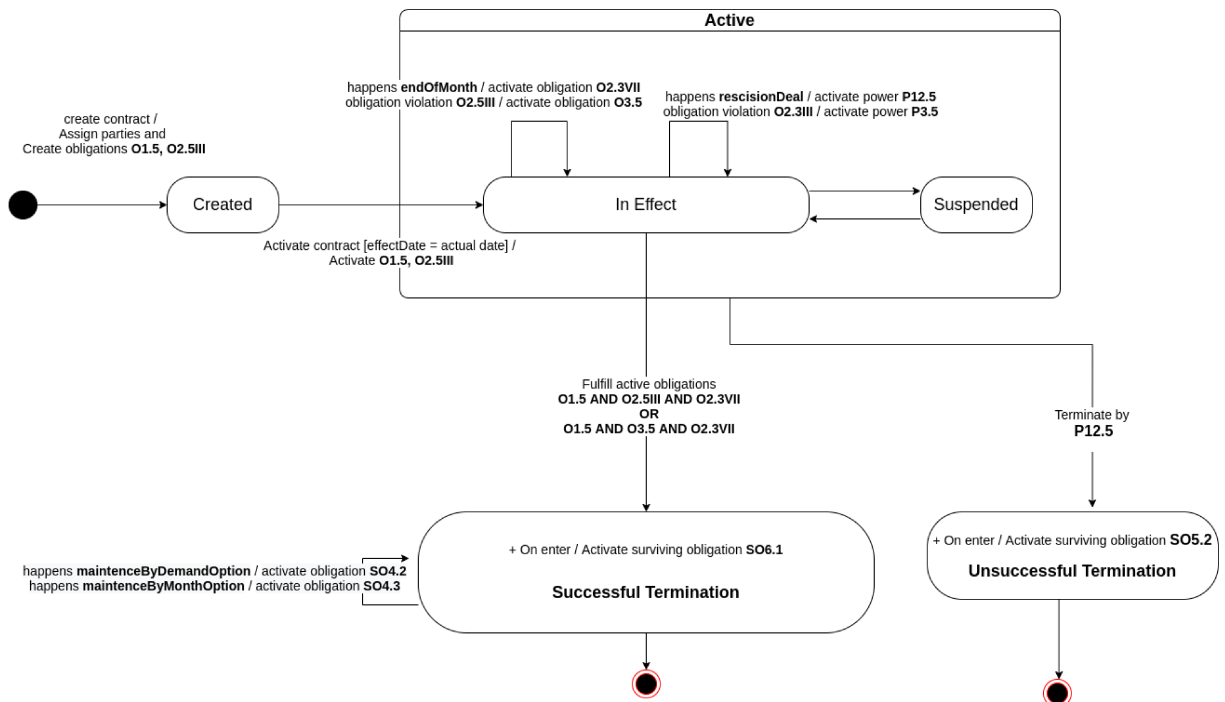
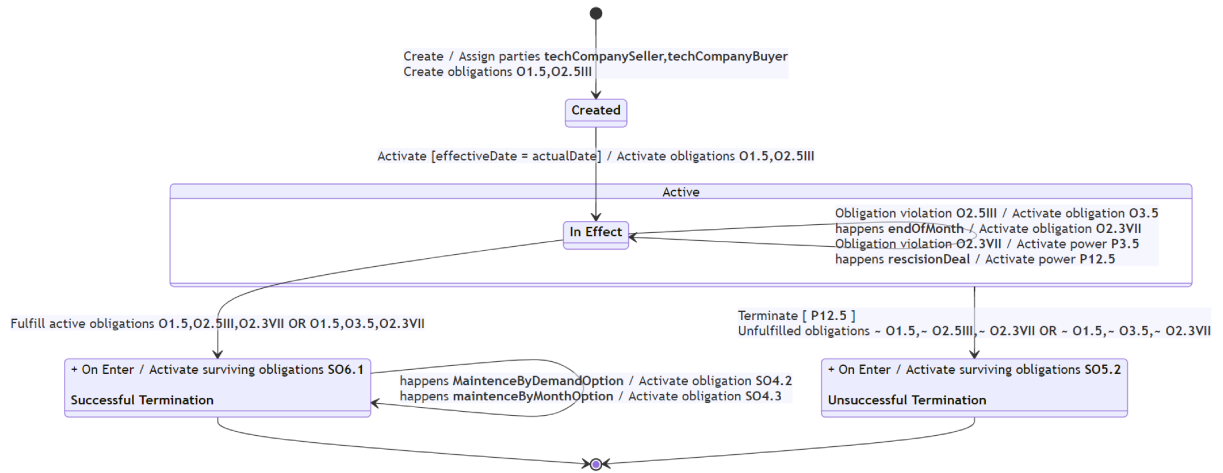


Figure 35 – Tech service automatically generated state diagram



5 EXPERIMENTS

At the outset, it is important to mention that, with a view to measuring whether state diagrams would facilitate the understanding of contracts, our initial intention was to have a first group exposed to a contract in natural language and the same contract represented in Symboleo, while a second group would be exposed to a contract in natural language, the same contract in Symboleo and its state diagram. However, a preliminary test with questions about the life cycle of a smart contract in Symboleo was applied to a single participant completely unknowledgeable about computing. It was noticed, however, that, besides the difficulty to provide the participant with a brief explanation of the language for the experiment, this participant was unable to understand Symboleo and the answers given were nothing but imprecise guesses. In this way, we realized that the use of Symboleo would not help and that the participants would end up looking for their answers only in the contract written in natural language. Therefore, our ultimate decision was to use the natural-language contract in comparison with the state diagram.

Consequently, in view of the above, to confirm whether state diagrams help to better understand legal contracts, two experiments were performed. The two experiments comprised a contract written in natural language and a state diagram corresponding to that same contract, and the participants were asked to answer questions about the clauses and the life cycle of said contract. In the first experiment, all the participants initially answered a first group of questions regarding only the contract written in natural language, and then regarding the contract with its corresponding state diagram in hand. In the second experiment, for the participants not to have a first contact with the contract when answering the second group of questions, two groups of participants were formed: one group answered the questions with access only to the contract in natural language; and the other group answered the questions with access to the contract and its corresponding state diagram. All responses were timed and only then were compared.

5.1 EXPERIMENT PROTOCOL

In both experiments, participants were invited according to their professional experience in the law and technology areas. Thus, in both areas there are participants who are still students and participants with more than 20 years of experience. Age and sex were not taken into account. Tables 5 and 6 show the profile details for each participant.

The hypothesis we want to prove with this experiment is the following: When we are asking questions about a specific legal contract, does the use of a state diagram representing the smart contract facilitate the understanding of this contract,

Table 5 – Participants' profile (Experiment 1)

Name	Area	Profession	Experience
B	Computer ence	Sci- Database Administra- tor and PhD Student	≈ 10 years
D	Computer ence	Sci- Professor and soft- ware engineer	≈ 10 years
F	Computer ence	Sci- Software engineer and PhD Student	≈ 5 years
I	Law	Lawyer	≈ 30 years
L	Computer ence	Sci- Software engineer	≈ 10 years
R	Law	Law student	No professional expe- rience

Table 6 – Participants' profile (Experiment 2)

Name	Area	Profession	Experience
B	Computer ence	Sci- Database Administra- tor and PhD Student	≈ 10 years
E	Computer ence	Sci- System analyst	≈ 10 years
D	Computer ence	Sci- Professor and soft- ware engineer	≈ 10 years
R	Law	Bachelor in Law and Social security ana- lyst	≈ 15 years
K	Law	Law student	No professional expe- rience
A	Computer ence	Sci- Project Manager	≈ 12 years
F	Computer ence	Sci- Software engineer and PhD Student	≈ 5 years
I	Law	Lawyer	≈ 30 years
L	Computer ence	Sci- Software engineer	≈ 10 years
R	Law	Law student	No professional expe- rience

resulting in a more objective and faster response?

5.2 EXPERIMENT 1

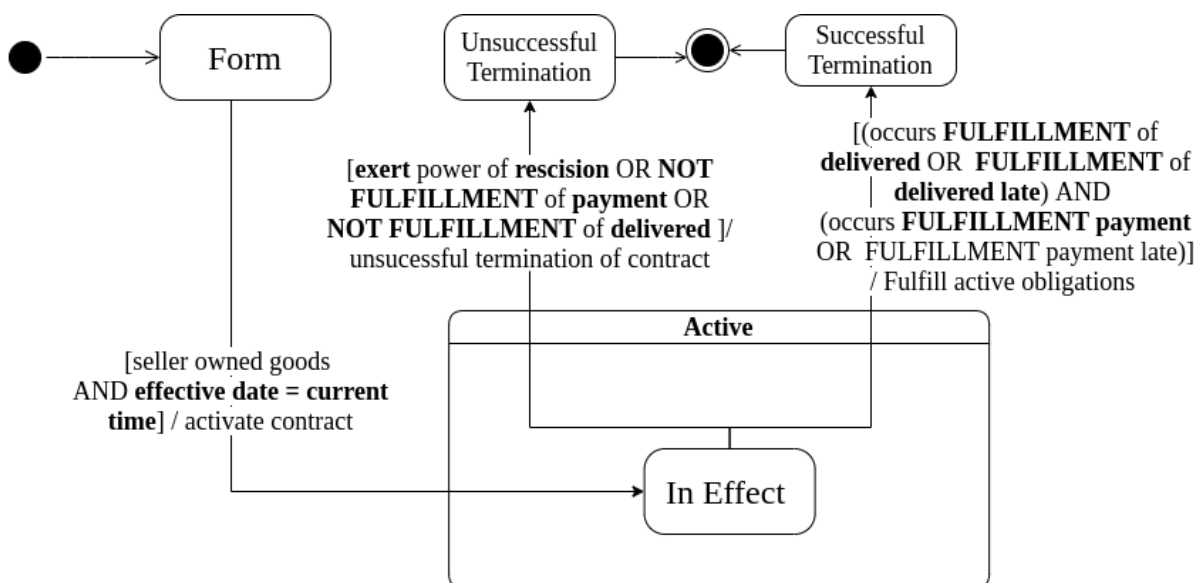
5.2.1 Overview

In the first experiment, six people were invited to participate, in which they were exposed to a contract for the purchase of paper sheets for a university. The participants were asked to answer eight questions about the functioning and flow of said contract. For the first four questions, they had only the contract written in natural language, and for the other four questions, they could use the corresponding diagram to help them. At each stage, they were asked to time how long it took them to answer each group of questions so that the time taken at each stage could be later compared.

The contract in question is available in Annex A, a simple purchase contract between two parties, a university and a paper and packaging company¹. The main clauses of the contract are the order request, payment due date, conditions for delivery of the ordered items, and payment methods. It also includes indemnity clauses and termination terms and conditions.

The diagram describes the states and transitions of the contract. Figure 36 shows the state diagram of the contract: the event necessary for the contract to come into effect, and the events to end it successfully or unsuccessfully.

Figure 36 – State diagram of the contract



¹ <https://repositorio.ufsc.br/bitstream/handle/123456789/193085/Contrato%20144.2019.pdf?sequence=118&isA>

As for the profile of the participants, four of them are computing professionals and know what a state diagram is, although some of them have little contact with this subject in their day-to-day lives. As for the other two participants, one is a lawyer and the other is a law student, and both had never had any contact with this type of diagram.

5.2.2 Execution

At the first stage, the participants were asked to answer the following questions:

1. Which obligations must be fulfilled in order for the payment obligation to be fulfilled?
2. What needs to be done, on the part of the seller, in case of delayed delivery?
3. What needs to happen before the buyer concludes the contract?
4. Can you identify which sets of obligations must be fulfilled (only the happy path) and in which order for the contract to end successfully?

And at the second stage:

1. What needs to happen for the contract to come into effect?
2. What needs to happen so that the seller needs to fulfill the obligation to pay indemnity?
3. What needs to happen for a successful mutual conclusion of the contract?
4. Can you identify what needs to happen for the contract to be terminated?

The participants were told that the answers could be generic and short. Nevertheless, each participant answered in their own way, some with more descriptive answers, others with more concise ones. The answers of the second stage were clearly more objective and more homogeneous among the participants, and were all right, in general.

Tables 7 and 8 contain the answers of the first and second groups to the above questions.

Table 7 – Participants' answers to the first group of questions

Name	Answer
B	<p>Q1: Obligations stipulated in the fifth clause of the contract. Q2: Notify the Principal within a maximum period of 48 hours prior to the delivery date. Inform the reasons with due evidence. Q3: Notify the Contractor 30 days in advance in the case of item 11.2.1; As for item 11.2.4, it is not clear. Q4: All requests must be made via email to the email address available on SICAF. Every request must be delivered within a period of up to 10 business days from the receipt of the request by the supplier. UFSC must make payments in accordance with the Public Bidding Notice. The Principal must comply with the obligations specified in Clause 4 and the Contractor in Clause 5. At the end of 12 months, the contract is concluded.</p>
D	<p>Q1: 7.5 Payment will be made by DCF within a maximum period of 30 days, counting from the receipt of materials/services and, as well as the delivery of the duly certified invoice/bill of sale. Q2: 5.6 Notify the Principal within a maximum period of 48 hours prior to the delivery date. Q3: 11.2.1 Determined by unilateral and written act of the Principal, in the cases listed in items I to XII. Also, 11.2.4 Determined by unilateral and written act of the Principal. Q4: From items 4.1 to 5.13, but the contract is not a flow.</p>
F	<p>Q1: Supply the agreed quantity of products; Fix any damage; Maintain the same conditions as those required for initial business approval; Maintain the Principal's requirements; Make available all means of attesting the quality of the materials; Report delivery problems 48 hours in advance; Be responsible for taxes, charges...; Be responsible for correct storage to protect the material; Do not outsource the service or part of it; Be responsible for transport; Ensure that goods do not contain toxic substances; Be responsible for the correct disposal and removal of waste at the place of delivery.</p> <p>Q2: Notify the Principal within a maximum period of 24 hours prior to the delivery date, the reasons that prevented delivery on the scheduled date, with due evidence. Q3: Notify the Contractor 30 days in advance. Q4: Sign contract; Delivery of services on time; Payments within 30 days after delivery of material/service and invoices for 12 months; After the last delivery, the last payment is made within 30 days, thus ending the contract.</p>

Name	Answer
I	<p>Q1: Delivery of purchased material and issuance of an invoice under the terms of the contract. Q2: Upon guaranteed prior defense, apply the sanctions of the Public Bidding Notice and termination. Q3: Advance notification (30 days), in the case of item 11.2.1 and balance of events already completed, list of payments made and still due, and indemnity and fines (items 11.3.1 to 11.3.3) Q4: Yes. Request received by the supplier, delivery of materials, receipt of invoice, payment.</p>
L	<p>Q1: The CONTRACTOR must schedule the delivery with the person responsible for receiving it; The products/materials must be delivered by the CONTRACTOR to the University during business hours from Monday to Friday; Meet the deadline of 10 working days for the delivery and inform, at least 48 hours in advance, the reasons that hindered meeting the deadline; the products must be delivered under the conditions and quantities described in the contract; Perishable products must have a shelf life of more than 6 months or more than half of the total period recommended by the manufacturer on the date of delivery; The PRINCIPAL must indicate "Approved" in writing on the invoice presented. Q2: Notify 48 hours in advance the reason for the delay; Submit a request for an extension of time by a formal letter. Q3: The CONTRACTOR does not deliver the products/materials within the stipulated period or outside the conditions and quantities described in the contract; The PRINCIPAL does not indicate "Approved" on the invoice presented. Q4: The PRINCIPAL provided the information and clarifications requested by the CONTRACTOR; The CONTRACTOR scheduled the delivery with the person responsible for receiving it; The PRINCIPAL was available for receipt on the scheduled delivery date; The CONTRACTOR delivered the products/materials under the conditions and quantities provided for in the contract; The PRINCIPAL indicated "Approved" on the invoice presented by the CONTRACTOR; The PRINCIPAL made the payment.</p>

Name	Answer
R	Q1: Supply the materials while maintaining the quality of materials. Q2: Notify the Principal within a maximum period of 48 hours about the reason that made compliance impossible. Q3: In case of unilateral termination, issue a notification 30 days in advance. Q4: It is according to what is set out in Clauses 4 and 5 of the current contract, where each party must fulfill their obligations, including delivery without delay and payment in accordance with the established terms.

Table 8 – Participants' answers to the second group of questions

Name	Answer
B	Q1: The seller owns the assets and the contract date = today's date. Q2: The seller delivers damaged products. Q3: After the contract is in effect, UFSC and Dicapel mutually agree to terminate the contract. Q4: When one of the parties exercises the unilateral right of termination or the Principal does not make the payment or the Contractor does not make the delivery.
D	Q1: Seller owned goods AND effective date = current date Q2: I was in doubt between "NO delay notification before 48 h until Due Date" or "NO payment of indemnity". Q3: UFSC and Dicapel terminate the contract mutually Q4: Exercise power of termination OR NOT FULFILLMENT of payment OR NOT FULFILLMENT of delivered.
F	Q1: When the seller has the products and the contract term is approaching. Q2: When any products are damaged. Q3: The Principal and the contractor agree to terminate the contract. Q4: When UFSC terminates the contract and does not pay the remaining amount; When one of the parties does not agree to terminate the contract; When UFSC does not make the payment (not even after the agreed date); When the contractor does not notify that it will not deliver 48 hours before the agreed time; At any time in the event of a termination.
I	Q1: Follow the stipulated date. Q2: Contractual provision and breach of the contract. Q3: Contractual provision and exercise of power by the parties. Q4: Yes, delivery failures and payment delays.
L	Q1: Creation of the contract; The CONTRACTOR has the products/materials; The current date is July 24, 2019. Q2: Creation of the contract; The contract enters into force; There is damage to any product/material. Q3: Creation of the contract; The contract enters into force; The CONTRACTOR and the PRINCIPAL agree amicably to terminate the contract. Q4: Creation of the contract; the contract enters into force; The CONTRACTOR does not make the delivery or the PRINCIPAL unilaterally exercises the power of termination or the PRINCIPAL does not make the payment.
R	Q1: First, the delivery of the product and later the payment. This way, the contract is closed. Q2: In case of delay or damage to the product to be delivered. Q3: When both companies mutually agree to conclude the contract. Q4: If payment or delivery is not made correctly.

Name	Answer
------	--------

For a final analysis, the participants were also asked if they really thought the diagram helped in any way with this specific question: “Did the experience of reading a contract through a state diagram make it easier for you to understand it and did it help you answer the questions?”. The answers are listed in Table 9.

Table 9 – Participants’ feedback on the use of the state diagram

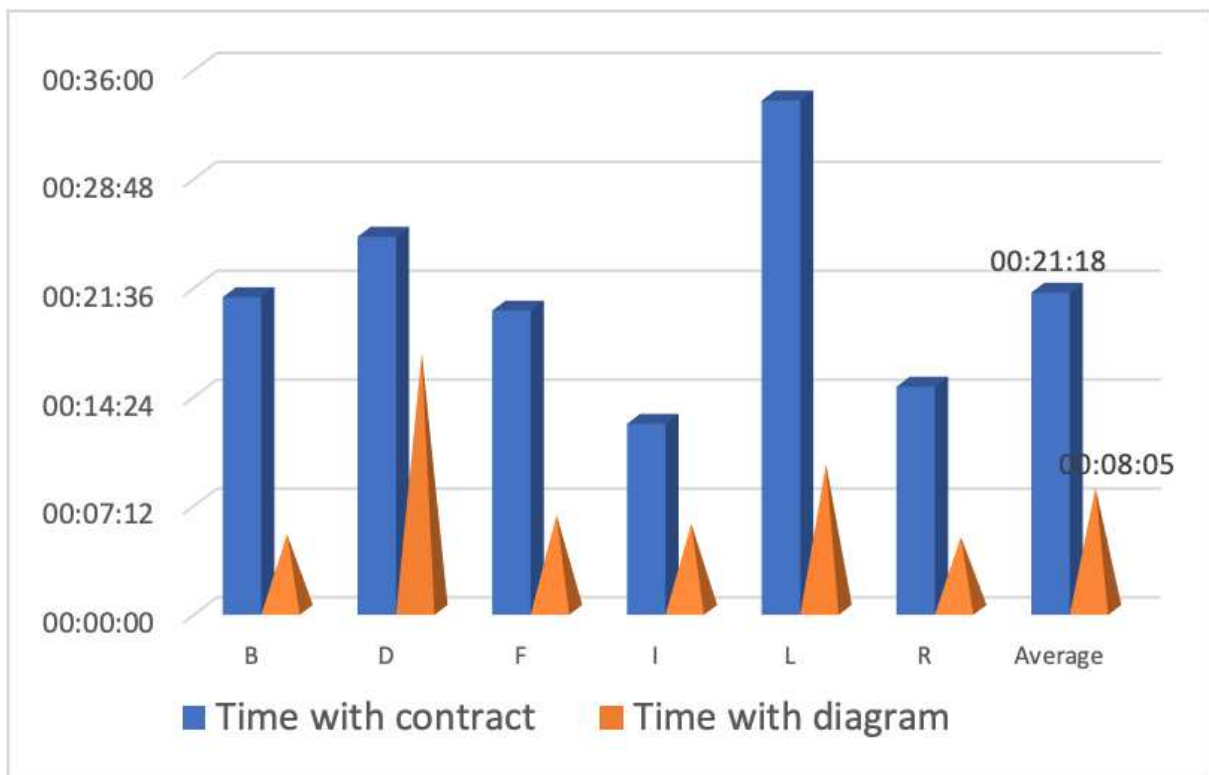
Name	Answer
ini-tial	
B	“It was much easier to use it indeed. Maybe it’d be interesting to complement it with more information. But if the goal is to analyze only the flow of events, it’s certainly much better.”
D	“The diagram systematizes the operation of the contract. In general, with just one reading of the contract I was unable to memorize and mentally systematize the content of the contract, especially details, so the diagram was very good to systematize it, thus making it easier to understand it. In general, the diagram is more objective, indicates a flow and is still indexed with a title. Certainly, the diagram helped me answer the questions. In my case it was a complement, because I read the contract before. I am not familiar with reading contracts, so, for a detailed understanding, it’s not clear yet, but the diagram helped me understand it.”
F	“Yes, it made it a lot easier for me.”
I	“Yes, by simplifying and highlighting what’s most relevant.”
L	“Yes, because it allows you to have a global view about the stages and events of the contract. When I answered the questions without using the diagrams, I had to read and reread some parts of the contract several times. Besides, without the diagrams, I would alternate between the contract chapters to build the order of events schematically in my mind. In practice, I had the impression that this mental process that I did in order to answer the questions in the first part was as if I had created in my mind the diagrams presented in the second part. When I started answering the questions of the second part, the one with the diagrams, it was much simpler, since I barely needed to go to the contract to identify the order of events, as the order was explicit in the diagrams themselves. Another difference between answering the questions with and without the diagrams is that in the second case I wrote much more than what was really necessary (considering that the answers in the second part are in fact correct).”
R	“Yes, it facilitated with the steps formulated, and described step by step, which reminds us of what was read in the contract.”

5.2.3 Results

The participants' answers in Table 9 attest that the diagram did facilitate the reading and understanding of the contract. As pointed out by some of them, the diagram systematized the operation of the contract and highlighted its most relevant points.

The graph in Figure 37 compares the time spent by each participant (the capital letters are the participants' name initials). The x-axis specifies the participants, and the y-axis contains the time spent in minutes. There are two bars for each participant: the time spent to answer the first four questions reading only the contract (rectangle) and the time spent to answer the last four questions reading both the contract and the state diagram (pyramid).

Figure 37 – Participants' time spent reading the contract and the diagram



Although this experiment comprised a very small number of participants and produced a subjective result, it is clear that the time spent to answer the questions was much shorter with the help of the state diagram. Figure 37 also shows the average time spent at each stage and highlights the expressive difference between the two situations.

Regarding correct or incorrect answers, on the first group of questions (first four questions), the majority of the answers were correct. Two of the six participants did not answer the first question as expected. Only one participant answered the

second question incorrectly. Also, only one participant answered the third question incorrectly, but all participants included unnecessary clauses of the contract. At last, the fourth question, all participants answered correctly, but with generic answers, like "All clauses inside clause fourth and fifth".

As for the second group of questions (last four questions), the majority of the answers were also correct. One of the six participants answered the first question incorrectly. For the second question, two participants answered incorrectly. For the third question, only one participant did not answer as expected, with a generic answer. And for the fourth question, all participants answered correctly.

On both stages of this experiment the majority of the answers were correct. But it is clear that for the second stage, the participants made less mistakes and the answers were more accurate.

5.2.4 Threats to validity

The main threat to the validity of the experiment is its small number of participants. Although the experiment sought to provide a majorly qualitative result, six participants are still a small number to assert that the diagram actually helped in the understanding of the contract.

Another important threat is that the same participants are in the two experiment phases. The fact that the participants had previously read the contract at the first stage may have accelerated their response time at the second stage, as pointed out by participant D in Table 9.

5.3 EXPERIMENT 2

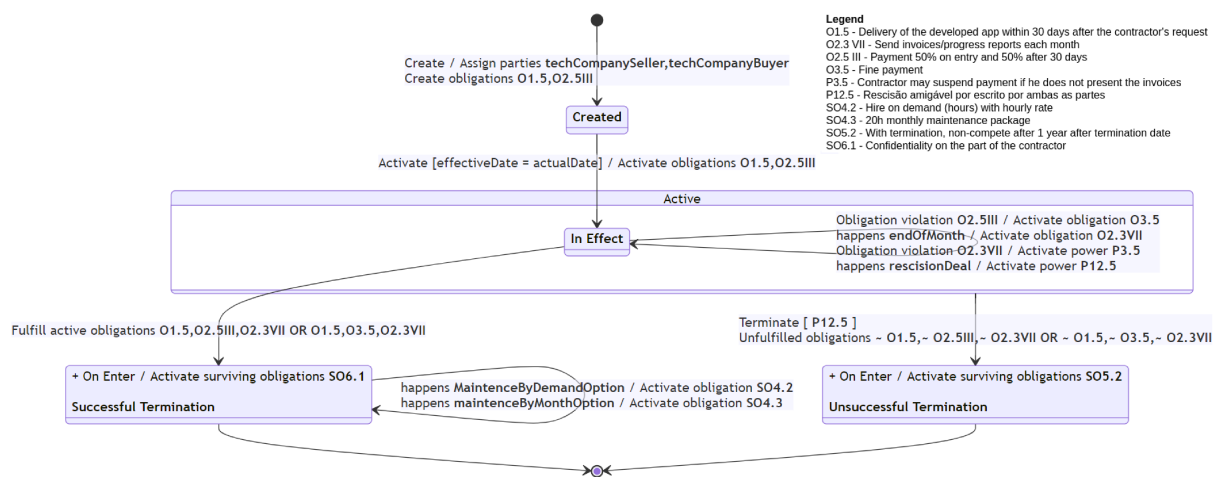
5.3.1 Overview

The second experiment was more elaborate. Ten people were invited to participate and were exposed to a legal contract signed by two companies regarding the provision of a software development service. The participants were asked to answer five questions about the functioning and flow of the contract. They were divided in two groups of five participants each, the first group with only the contract written in natural language to help responding the questions, and the second group two with the contract written in natural language and the corresponding state diagram. The participants were also asked to time how long they took to answer each question.

The contract in question is available in Annex B. The main clauses of the contract are the software order with a specific scope, payment due date, and conditions for delivery of the ordered service. It also includes indemnity clauses and termination terms and conditions.

The state diagram describes the states and transitions of the contract, showing the event necessary for the contract to come into effect, and the events to end it successfully or unsuccessfully. The JSON representing this state diagram was automatically generated by the algorithm described above, available in Appendix B. Figure 38 contains the specific state diagram of the contract that was generated with the algorithm and used in the experiment.

Figure 38 – State diagram of the legal contract



The profile of the participants remains the same, that is, six computing professionals who know what a state diagram is, despite having little contact with this subject; and the other four participants are law professionals who had never had any contact with this type of diagram before.

5.3.2 Execution

The following five questions were asked to all participants:

1. What obligations do you think must be fulfilled for the contract to end successfully (as expected), without any setbacks (delays, for example)?
2. If the Contractor does not comply with the obligation described in clause 2.3 VII, what action can the Principal take?
3. What clauses related to the maintenance of the app can be activated after the end of the contract?
4. What action or actions can be taken by the parties to terminate the contract, that is, to end it without success?
5. There is a clause that must be fulfilled by the Contractor when the contract is completed WITHOUT success. What is it?

The participants were told that the answers could be generic and short. Nevertheless, each participant answered in their own way, some with more descriptive answers, others with more concise ones. In general, the answers of second group (with the state diagram) were clearly more objective and more homogeneous among the participants, and the majority of them were correct. Tables 10 and 11 show the full answer of each participant. The first column specifies the participant, the second column the professional area, where "CS" stands for Computer Science and "L" stands for Law.

Table 10 – Participants' answers without the state diagram

Name	Area	Answer
B	CS	Q1: Everything specified in Clauses 1,2,3,4 and 5. Q2: Payment will be delayed, and interest free. Q3: 4.2 and 4.3. Q4: Only by written agreement of both parties. Q5: 5.2
E	CS	Q1: The object of the contract must be delivered within the established period (up to 45 days from the signing of the contract). Q2: Fine of 2% (two percent), in addition to monetary restatement by the INPC price correction index established in this contract, "pro rata day", 4/8 plus interest of 1% (one percent) per month, calculated on the debt amount already corrected. Q3: The PRINCIPAL is responsible, but the CONTRACTOR can be convened at a cost of BRL 180.00 per hour, or by contracting a 20-hour package at a cost of BRL 120.00 per hour. Q4: If the CONTRACTOR finds that it is impossible to provide services remotely, a visit to the PRINCIPAL's main office or branches will be scheduled, upon reimbursement of travel, accommodation and food expenses by the PRINCIPAL with prior approval by the PRINCIPAL. Q5: If the CONTRACTOR finds that it is impossible to provide services remotely, a visit to the PRINCIPAL's headquarters or branches will be scheduled, upon reimbursement of travel, accommodation and food expenses by the PRINCIPAL with prior approval by the PRINCIPAL.

Name	Area	Answer
D	CS	<p>Q1: When items 2.3 to 3.5 are not disrespected. Q2: I did not explicitly identify the resolution and what is made by the PRINCIPAL if item VII of clause 2.3 is not complied with by the CONTRACTOR. Q3: CLAUSE 4 - TERM AND TERMINATION; I was in doubt about which period these are valid: CLAUSE 5 - NON-COMPETE; CLAUSE 6 - CONFIDENTIALITY. Q4: This was not clear to me. Implicitly, if the parties violate any clause. Above all, item 12.5 mentions: This contract may not be assigned, in whole or in part, except for the express and written agreement of both parties. Q5: 5.2. If the contract is terminated before the end of the term of this partnership, the CONTRACTOR undertakes to respect this Clause for a period of 1 (one) year from the effective date of termination.</p>
R	L	<p>Q1: Delivery of the object under the terms and conditions stipulated in the contract, as well as the fulfillment of the obligations assumed by both parties. Q2: Based on § 12a, item 12.1, there may be tolerance by the Principal, not giving rise to novation, contacting the Contractor via email, requesting the documentation required for payment. Q3: §2a, item 2.3, III and VI - §5a, item. 5.1 — §6a, item 6.1 and 6.2 Sole §. Q4: If by mutual agreement between the parties in writing, undertake to respect the non-compete clause for a period of 1 (one) year from the effective date of termination. Q5: §5a, item 5.2.</p>

Name	Area	Answer
K	L	Q1:: The product must be delivered by the Contractor within the specified period, according to Clause 1.5, the Contractor must provide all necessary data and information, as stipulated in Clause 2.5. In addition, payment for the provision of service must be made within the term and in the amount determined in Clause 3.1 and 3.2. Q2: I did not find any clauses in the contract regarding its termination, however the Principal may judicially request the termination of the contract. Q3: 4.2 and 4.3. Q4: When the Principal does not provide all the information necessary for the preparation, the Contractor does not deliver the product and the Principal does not make the payment as determined in the contract. Q5: Clause 5.2.

Table 11 – Participants' answers with the state diagram

Name	Area	Answer
A	CS	Q1: Delivery in 30 days, down payment of 50% and 50% on deliver, and issuance of invoices each month. Q2: Can suspend payment. Q3: Hire by demand, hire 20h per month for maintance. Q4: Amicable termination in written agreement. Q5: No competition after 1 year.
F	CS	Q1: Delivery after 30 days of Principal's request, down payment 50% plus 50% after 30 days, and issuance of invoices or delivery after 30 days of Principal's request, payment of fine and issuance of invoices. Q2: Payment of fine. Q3: SO4.2 or SO4.3. Q4: Amicable termination in written agreement by both parties. Q5: SO5.2.
I	L	Q1: All necessary information will be provided by the Principal to the Contractor. Q2: Request shipment and delivery of what is provided for in item 3.5. Q3: Clauses 4a and 5a. Q4: Did not find it. Q5: 5.2.
L	CS	Q1: Clauses 1.5 and 2.3VII and 2.5III. Q2: Clause 3.5. Q3: Clauses 4.2 and 4.3. Q4: Clause 12.5. Q5: Clause 5.2
R	L	Q1: Delivery of application after 30 days of contract. Q2: Principal may suspend if not submit notes. Q3: Clause 4.2. Q4: With termination. Q5: Clause 5 - non-compete after 1 year of termination.

Name	Area	Answer
------	------	--------

For a final analysis, the participants of the second group (with the state diagram) were also asked if they really thought the state diagram helped in any way: “Do you think that the state diagram actually helped you understand the contract and find the answers?”. Their answers are listed in Table 12.

Table 12 – Participants’ feedback on the use of the state diagram

Participant Answer	
A	“yes, after understanding how it works, it was simple to find the answers.”
L	“Sure, the answers were basically given by reading the diagram.”
F	“Yes, it helped a lot.”
I	“Yes.”
R	“Yes.”

5.3.3 Results

The graph in Figure 39 compares the time spent by each participant in each question (bars). The x-axis specifies each participant (including their area of expertise – CS for Computer Science and L for Law; and whether or not they used the state diagram), and the y-axis specifies the time spent in minutes.

The first five participants in the graph shown in Figure 39 (in blue) are those who had the diagram to help them, and it is clear that they spent less time answering the five questions. Despite that, the majority of participants, even the ones with the state diagram, spent more time to answer the first question, and it is probably because they took some time for an overall glance at the contract.

The graph in Figure 40 has a simplified comparison of the total time spent by the participants with and without the state diagram. It shows clearly that the time spent by the participants using the state diagram is actually shorter. The average time spent by the participants that had the state diagram was 09’23” while the average time spent by those that had only the contract written in natural language was 26’11”. In view of that, we can conclude that the state diagram representing the legal contract proved very helpful.

In general, the computer science experts took less time to answer the questions, with or without the diagram, and it may indicate that the diagram should be more self-explanatory and provide simpler information for laypeople. Even though the diagram considerably helped, as evinced by the time-spent comparison, the assertive responses and the participants’ feedback, it was also observed that some participants remained dependant on the written contract.

Figure 39 – Participants’ time spent reading the contract and the state diagram for each question

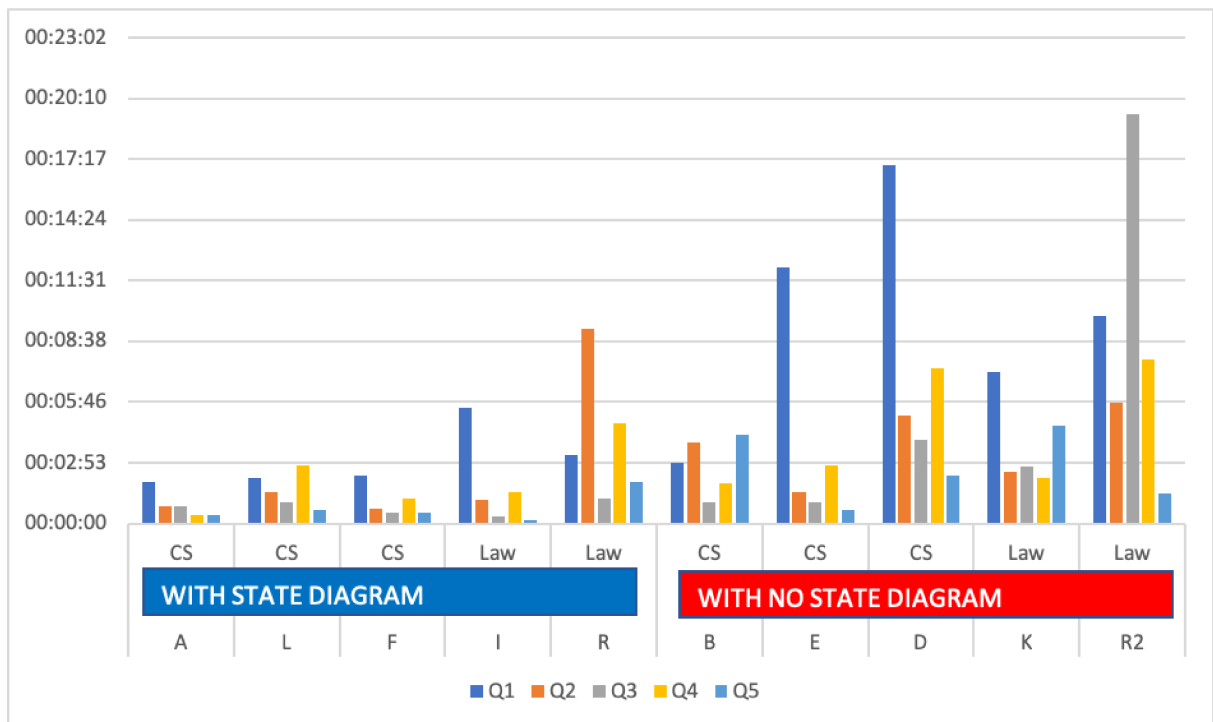
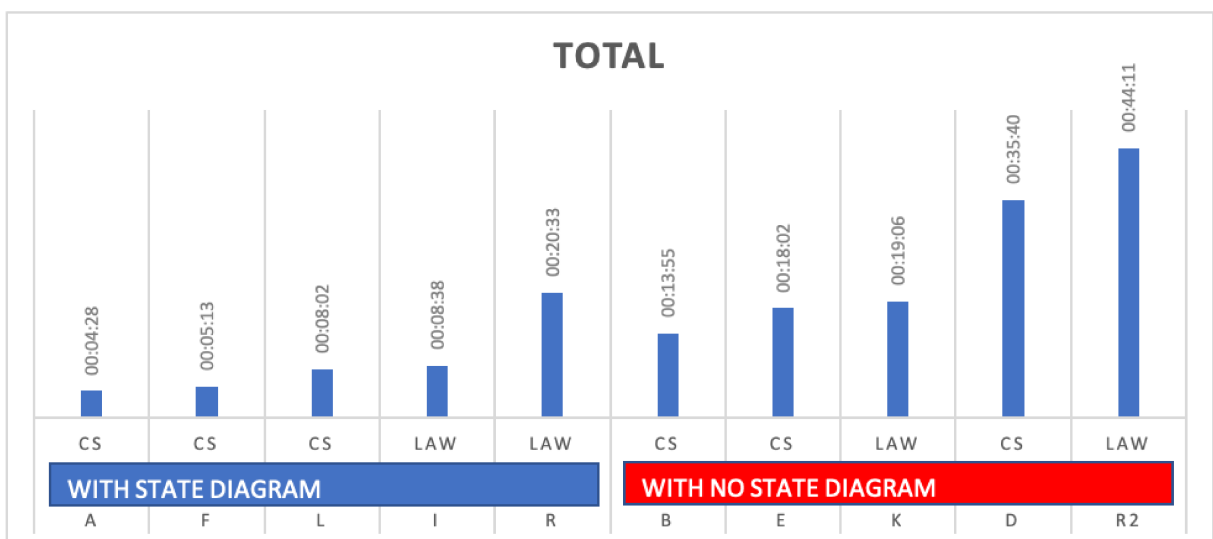


Figure 40 – Participants’ total time spent reading the contract and the state diagram



Regarding the analysis of the correctness of the answers, it is possible to observe that there were a consider number of mistakes, mainly for the group that did not have the diagram for help.

As for the first group of participants (five participants with state diagram), there were more correct answers, mainly for the computer science experts. For the first question, one of them answered incorrectly and another one gave an incomplete answer (one clause was missing). For the second question, there were two incorrect answers. For the third question, both law experts gave incomplete and generic answers. For the fourth question, only one participant did not know the answer (did not find it). And for the fifth question, all participants answered correctly.

Then, for the second group of participants (five participants with no state diagram), for the first question, two of the participants aswered correctly (both of law area). For the second question, only one participant answered correctly, other two participants said that they did not know the answer. For the third question, three participants answered correctly and one of them said he did not know the answer. For the fourth question, two participants answered correctly, one of them explained that it was not clear on the contract text, but included the right answer. As for the last question, only one of the participants answered incorrectly.

In this second experiment, it is easy to see that the first group of answers were more accurate. But it was also possible to observe that the majority of the mistakes were from the law experts, only on the third question a computer science expert made a mistake. And this is possibly related to lack of familiarity with state diagrams. In addition, these results show that the state diagram may not be so simple and easy to understand for non-technical users.

5.3.4 Threats to validity

The main threat to the validity of the experiment is its small number of participants. Even though the experiment produced a majorly qualitative result, ten participants are still a small number to confirm that state diagrams actually help in the understanding of legal contracts.

For a future experiment, our intention is that all participants, at both stages, be exposed to different contracts: they would answer questions from one legal contract using only the contract written in natural language and also answer questions from another legal contract along with its respective state diagram. This way, we would be able to compare the performance of a participant either using or not the state diagram.

The legal contract used in this experiment was transformed into formal language manually, without any tools that could guarantee correctness, so this is a shortcoming as regards the creation of the diagram. Important information may have

been misstated or even left out.

Another threat to validity was the lack of experts in smart contracts as part of the participants. Experts in smart contracts could have a different opinion about the use of state diagrams to better understand smart contracts specified using a domain-specific language as *Symboleo*.

Another future experiment would be the implementation of a smart contract by developers using the generated state diagram. In this way, it will also be possible to analyze whether state diagrams could also facilitate the creation of smart contracts.

6 CONCLUSIONS

In the present master's thesis, we investigated the use of state diagrams to represent legal smart contracts. Over the period of investigation, the present study produced three important contributions: (1) a systematic mapping to find out how state diagrams are being used to represent smart contracts; (2) the proposal and implementation of an algorithm to automate the creation of state diagrams that represent smart contracts specified with the domain-specific language *Symboleo*; and (3) two experiments demonstrating that state diagrams can facilitate the understanding of legal contracts written in natural language.

The first contribution, i.e., the systematic mapping, allowed us to observe that several studies have represented smart contracts as state diagrams and they usually do that for reasons like model verification, correctness and improvement of the smart contract design. But we observed that there is no pattern or consensus in the literature about how to map a legal smart contract into states and transitions of a state diagram. The systematic mapping also allowed us to observe that smart contracts are a subject that has truly become popular and has drawn the attention of people from some technical areas other than computing (such as business and law). Such popularity and interdisciplinarity are some of the motivations for the quest for easiness in the writing of smart contracts.

Among the related work retrieved in the systematic mapping, many of them contributed to the present study, as they: (1) contained the mapping of states of a smart contract, in almost all of them (18), as in (BAI et al., 2018), (GOVERNATORI et al., 2018), (XU; FINK, 2020), (BANACH, 2020), (SKOTNICA; PERGL, 2020), (MAVRIDOU et al., 2019), (MAVRIDOU; LASZKA, 2018), (LADLEIF; WESKE, 2019a), (PINNA; IBBA, 2018), (SHARIFI, S. et al., 2020), (WEINGAERTNER et al., 2018), (ALQAHTANI et al., 2020), (ELIZONDO et al., 2019), (BOOGAARD, 2018), (HE, Xudong, n.d.), (BERTOLINI, 2020), (GARAMVÖLGYI et al., 2018) and (MITCHELL, n.d.); and (2) presented considerations of legal aspects, as in (GOVERNATORI et al., 2018), (XU; FINK, 2020), (LADLEIF; WESKE, 2019a), (SHARIFI, S. S., 2020) and (WEINGAERTNER et al., 2018). Two other aspects that were taken into account as distinguishing features of the present study regard automated code generation, considering that, even though our aim of this study did not encompass automated code generation, it is a good reference for a better understanding of the inputs and the standards recognized for generating smart contracts, for example, as can be seen in (MADL et al., 2019), (MAVRIDOU et al., 2019), (MAVRIDOU; LASZKA, 2018) and (CONCHON; KORNEVA; ZAÏDI, 2020).

Another contribution of this study is an algorithm, along with its implementation, for the automated creation of state diagrams that represent smart contracts specified in a domain-specific language for smart contracts called *Symboleo*. As other

formal models, Symboleo is also used to provide a systematic method to create and deploy smart contracts, in an attempt to achieve easiness in writing smart contracts or/and their correctness. However, Symboleo, as any other formal model, is not easy to be understood by users. On the other hand, state diagrams can assist in the modeling of smart contracts, thus enabling the understanding of legal contracts.

One last contribution was the two experiments conducted to evaluate the understanding of legal contracts written in natural language alongside a state diagram. The experiment yielded subjective results, however they lead to a conclusion that state diagrams can facilitate the understanding of legal contracts.

6.0.1 Future Work

As further work, we intend to include the mapping of other smart-contract domain languages that have in their specifications the definition of obligations and powers, allowing these languages, besides Symboleo, to be received as input to the implemented algorithm. To ensure the correctness of the generated state diagrams, we want to create a formal prove for the transformation algorithm. We also want to use diagrams in the future as a way of verifying the life cycle of specified contracts, ensuring the correctness of state transitions. Additionally, we seek to improve these experiments, also using a specified contract with a domain-specific language, rather than just the contract in natural language, also increasing the number of participants and the number of questions. We also seek to realize another experiment with smart contract developers, asking them to implement a smart contract using the generated state diagram as a support.

REFERENCES

ABDELHAMID, Manar; HASSAN, Ghada. Blockchain and Smart Contracts. In: ACM. PROCEEDINGS of the 2019 8th International Conference on Software and Information Engineering. [S.l.: s.n.], 2019. P. 91–95.

ALQAHTANI, Sarra; HE, Xinchu; GAMBLE, Rose; MAURICIO, Papa. Formal Verification of Functional Requirements for Smart Contract Compositions in Supply Chain Management Systems. In: PROCEEDINGS of the 53rd Hawaii International Conference on System Sciences. [S.l.]: Hawaii International Conference on System Sciences, 2020.

AMARAL DE SOUSA, Victor; BURNAY, Corentin; SNOECK, Monique. B-MERODE: A Model-Driven Engineering and Artifact-Centric Approach to Generate Smart Contracts. In.

AMATO, Flora; COZZOLINO, Giovanni; MOSCATO, Francesco; MOSCATO, Vincenzo; XHAFI, Fatos. A Model for Verification and Validation of Law Compliance of Smart Contracts in IoT Environment. **IEEE Transactions on Industrial Informatics**, v. 17, n. 11, p. 7752–7759, Nov. 2021. ISSN 1941-0050.

AMBLER, Scott W. [S.l.]: Ambysoft Inc., 2003. Available from:
<http://www.agilemodeling.com/artifacts/stateMachineDiagram.htm>.

BAI, Xiaomin; CHENG, Zijing; DUAN, Zhangbo; HU, Kai. Formal Modeling and Verification of Smart Contracts. In: PROCEEDINGS of the 2018 7th International Conference on Software and Computer Applications. Kuantan, Malaysia: Association for Computing Machinery, 2018. (ICSCA 2018), p. 322–326.

BANACH, R. Verification-Led Smart Contracts. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, 11599 LNCS, p. 106–121, 2020. cited By 0.

BARESI, L.; GARZOTTO, F.; PAOLINI, P. Extending UML for modeling Web applications. In: PROCEEDINGS of the 34th Annual Hawaii International Conference on System Sciences. [S.l.: s.n.], 2001. 10 pp.-.

BASHIR, Imran. **Mastering blockchain**. [S.l.]: Packt Publishing Ltd, 2017.

BERTOLINI, MARCELLO. Enforcing commitments with blockchain: an approach to generate smart contracts for choreographed business processes, 2020.

BODEVEIX, Jean-Paul; FILALI, Mamoun; LAWALL, Julia; MULLER, Gilles. Formal Methods Meet Domain Specific Languages. In: ROMIJN, Judi; SMITH, Graeme; POL, Jaco van de (Eds.). **Integrated Formal Methods**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. P. 187–206.

BOOGAARD, Kees. **A model-driven approach to smart contract development**. 2018. MA thesis.

CHANG, Shuchih Ernest; LUO, Hueimin Louis; CHEN, YiChian. Blockchain-enabled trade finance innovation: A potential paradigm shift on using letter of credit. **Sustainability**, Multidisciplinary Digital Publishing Institute, v. 12, n. 1, p. 188, 2020.

CHEN, E; QIN, Bohan; ZHU, Yan; SONG, Weijing; WANG, Shengdian; CHU, William Cheng-Chung; YAU, Stephen S. SPESC-Translator: Towards Automatically Smart Legal Contract Conversion for Blockchain-based Auction Services. **IEEE Transactions on Services Computing**, p. 1–1, 2021. ISSN 1939-1374.

CONCHON, Sylvain; KORNEVA, Alexandrina; ZAÏDI, Fatiha. Verifying Smart Contracts with Cubicle. In: LECTURE Notes in Computer Science. [S.l.]: Springer International Publishing, 2020. P. 312–324.

DIXIT, Abhishek; DEVAL, Vipin; DWIVEDI, Vimal; NORTA, Alex; DRAHEIM, Dirk. Towards user-centered and legally relevant smart-contract development: A systematic literature review. **Journal of Industrial Information Integration**, v. 26, p. 100314, 2022. ISSN 2452-414X.

ECMA. **The JSON data interchange syntax**. [S.l.: s.n.], 2017. Available from: <https://www.ecma-international.org/publications-and-standards/standards/ecma-404/>.

EDMONDS, Jeff. **How to think about algorithms**. [S.l.]: Cambridge University Press, 2008.

ELIZONDO, Sergio; WATTAM, Stephen; ROBU, Valentin; JONES, Rachel; OAKES, Graham. **A Smart Contracting Framework for Aggregators of Demand-Side Response**. [S.l.]: AIM, 2019.

FLOOD, M.D.; GOODENOUGH, O.R. Contract as automaton: representing a simple financial agreement in computational form. **Artificial Intelligence and Law**, 2021. cited By 1.

GARAMVÖLGYI, P.; KOCSIS, I.; GEHL, B.; KLENIK, A. Towards Model-Driven Engineering of Smart Contracts for Cyber-Physical Systems. In: 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W). [S.l.: s.n.], June 2018. P. 134–139.

GATTESCHI, Valentina; LAMBERTI, Fabrizio; DEMARTINI, Claudio; PRANTEDA, Chiara; SANTAMARÍA, Víctor. Blockchain and Smart Contracts for Insurance: Is the Technology Mature Enough? **Future Internet**, MDPI AG, v. 10, n. 2, p. 20, Feb. 2018. ISSN 1999-5903.

GOVERNATORI, Guido; IDELBERGER, Florian; MILOSEVIC, Zoran; RIVERET, Régis; SARTOR, Giovanni; XU, Xiwei. On legal contracts, imperative and declarative smart contracts, and blockchain systems. **Artificial Intelligence and Law**, Springer, v. 26, n. 4, p. 377–409, 2018.

HE, Xiao; QIN, Bohan; ZHU, Yan; CHEN, Xing; LIU, Yi. Spesc: A specification language for smart contracts. In: IEEE. 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC). [S.l.: s.n.], 2018. v. 1, p. 132–137.

HE, Xudong. Modeling and Analyzing Smart Contracts using Predicate Transition Nets.

HYPERLEDGER. **Smart Contracts and Chaincode**. [S.l.: s.n.], 2020. Available from: <https://hyperledger-fabric.readthedocs.io/en/latest/smartcontract/smartcontract.html>.

IDELBERGER, Florian; GOVERNATORI, Guido; RIVERET, Régis; SARTOR, Giovanni. Evaluation of logic-based smart contracts for blockchain systems. In: SPRINGER. INTERNATIONAL symposium on rules and rule markup languages for the semantic web. [S.l.: s.n.], 2016. P. 167–183.

JURGELAITIS, M.; CEPONIENE, L.; BUTKIENE, R. Solidity Code Generation From UML State Machines in Model-Driven Smart Contract Development. **IEEE Access**, v. 10, p. 33465–33481, 2022. cited By 1.

LADLEIF, Jan; WESKE, Mathias. A Legal Interpretation of Choreography Models. In: DI FRANCESCOMARINO, Chiara; DIJKMAN, Remco; ZDUN, Uwe (Eds.). **Business Process Management Workshops**. Cham: Springer International Publishing, 2019. P. 651–663.

LADLEIF, Jan; WESKE, Mathias. A Unifying Model of Legal Smart Contracts. In: LAENDER, Alberto H. F.; PERNICI, Barbara; LIM, Ee-Peng; OLIVEIRA, José Palazzo M. de (Eds.). **Conceptual Modeling**. Cham: Springer International Publishing, 2019. P. 323–337.

LAURENCE, Tiana. **Blockchain for dummies**. [S.l.]: John Wiley & Sons, 2019.

LUU, Loi; CHU, Duc-Hiep; OLICKEL, Hrishi; SAXENA, Prateek; HOBOR, Aquinas. Making smart contracts smarter. In: PROCEEDINGS of the 2016 ACM SIGSAC conference on computer and communications security. [S.l.: s.n.], 2016. P. 254–269.

MADL, G.; BATHEN, L.; FLORES, G.; JADAV, D. Formal Verification of Smart Contracts Using Interface Automata. In: 2019 IEEE International Conference on Blockchain (Blockchain). [S.l.: s.n.], 2019. P. 556–563.

MANAV, Gupta. **Blockchain for dummies**. [S.l.]: Hoboken: John Wiley & Sons, Inc, 2017.

MARCHESI, Michele; MARCHESI, Lodovica; TONELLI, Roberto. An Agile Software Engineering Method to Design Blockchain Applications. In: PROCEEDINGS of the 14th Central and Eastern European Software Engineering Conference Russia. Moscow, Russian Federation: Association for Computing Machinery, 2018. (CEE-SECR '18).

MAVRIDOU, Anastasia; LASZKA, Aron. Designing Secure Ethereum Smart Contracts: A Finite State Machine Based Approach. In: MEIKLEJOHN, Sarah; SAKO, Kazue (Eds.). **Financial Cryptography and Data Security**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2018. P. 523–540.

MAVRIDOU, Anastasia; LASZKA, Aron; STACHTIARI, Emmanouela; DUBEY, Abhishek. VeriSolid: Correct-by-design smart contracts for Ethereum. In: SPRINGER.

INTERNATIONAL Conference on Financial Cryptography and Data Security. [S.l.: s.n.], 2019. P. 446–465.

MERNIK, Marjan; HEERING, Jan; SLOANE, Anthony M. When and How to Develop Domain-Specific Languages. **ACM Comput. Surv.**, Association for Computing Machinery, New York, NY, USA, v. 37, n. 4, p. 316–344, Dec. 2005. ISSN 0360-0300.

MITCHELL, Ian. Blockchain Medicine Administration Records (BMAR): Reflections and Modelling Blockchain with UML.

NARAYANAN, Arvind; MILLER, Andrew. Research for Practice: Cryptocurrencies, Blockchains, and Smart Contracts. In: IEEE Symposium on Security and Privacy. [S.l.: s.n.], 2016.

OMG. **OMG Unified Modeling Language (OMG UML)**. [S.l.: s.n.], 2017. Available from: <https://www.omg.org/spec/UML/2.5.1/PDF>.

PETERSEN, Kai; FELDT, Robert; MUJTABA, Shahid; MATTSSON, Michael. Systematic mapping studies in software engineering. In: 12TH International Conference on Evaluation and Assessment in Software Engineering (EASE) 12. [S.l.: s.n.], 2008. P. 1–10.

PETERSEN, Kai; VAKKALANKA, Sairam; KUZNIARZ, Ludwik. Guidelines for conducting systematic mapping studies in software engineering: An update. **Information and software technology**, Elsevier, v. 64, p. 1–18, 2015.

PINNA, Andrea; IBBA, Simona. A Blockchain-Based Decentralized System for Proper Handling of Temporary Employment Contracts. In: ADVANCES in Intelligent Systems and Computing. [S.l.]: Springer International Publishing, Nov. 2018. P. 1231–1243.

SHARIFI, Sepehr; PARVIZIMOSAED, Alireza; AMYOT, Daniel; LOGRIPPO, Luigi; MYLOPOULOS, John. A Specification Language for Smart Contracts. In: PROCEEDINGS of the 28th IEEE Requirements Engineering Conference (RE'20). Zurich: [s.n.], 2020.

SHARIFI, Seyed Sepehr. **Smart Contracts: From Formal Specification to Blockchain Code**. 2020. PhD thesis – Université d'Ottawa/University of Ottawa.

SKOTNICA, Marek; PERGL, Robert. Das Contract - A Visual Domain Specific Language for Modeling Blockchain Smart Contracts. In: AVEIRO, David; GUIZZARDI, Giancarlo;

BORBINHA, José (Eds.). **Advances in Enterprise Engineering XIII**. Cham: Springer International Publishing, 2020. P. 149–166.

SOURCEMAKING. **Design patterns and refactoring**. [S.l.: s.n.], 2007. Available from: <https://sourcemaking.com/uml/modeling-it-systems/the-behavioral-view/the-life-of-an-object>.

SZABO, Nick. Formalizing and Securing Relationships on Public Networks. **First Monday**, v. 2, n. 9, Sept. 1997.

VILAIN, Patricia. **Using Acceptance Tests as part of the Requirements Specification of Smart Contracts**. [S.l.], Dec. 2021. P. 19.

WEINGAERTNER, Tim; RAO, Rahul; ETTLIN, Jasmin; SUTER, Patrick; DUBLANC, Philipp. Smart Contracts Using Blockly: Representing a Purchase Agreement Using a Graphical Programming Language. In: 2018 Crypto Valley Conference on Blockchain Technology (CVCBT). [S.l.]: IEEE, June 2018.

WEISS, Michael. XML Metadata Interchange. In: **Encyclopedia of Database Systems**. Ed. by LING LIU and M. TAMER ÖZSU. Boston, MA: Springer US, 2009. P. 3597–3597. ISBN 978-0-387-39940-9.

WOHRER, M.; ZDUN, U. From Domain-Specific Language to Code: Smart Contracts and the Application of Design Patterns. **IEEE Software**, v. 37, n. 5, p. 37–42, Sept. 2020. ISSN 1937-4194.

WÖHRER, Maximilian; ZDUN, Uwe. Domain Specific Language for Smart Contract Development. In: IEEE International Conference on Blockchain and Cryptocurrency. [S.l.: s.n.], 2020.

XU, Weifeng; FINK, Glenn A. Building Executable Secure Design Models for Smart Contracts with Formal Methods. In: BRACCIALI, Andrea; CLARK, Jeremy; PINTORE, Federico; RØNNE, Peter B.; SALA, Massimiliano (Eds.). **Financial Cryptography and Data Security**. Cham: Springer International Publishing, 2020. P. 154–169.

GLOSSARY

APPENDIX A - LITERATURE REVIEW - ARTICLES' TITLES

Table 13 – Articles select in the literature review

Library	Title	Selected in step			Selected
		1	2	3	
1 ACM	Automatic Smart Contract Generation Using Controlled Natural Language and Template (Tateishi et al.)	X	X		
2 ACM	An Agile Software Engineering Method to Design Blockchain Applications (Marchesi et al.)	X	X	X	X
3 ACM	DeLottery: A Novel Decentralized Lottery System Based on Blockchain Technology (Jia et al.)	X	X		
4 ACM	Formal Modeling and Verification of Smart Contracts (Bai et al.)	X	X	X	X
5 ACM	On Legal Contracts, Imperative and Declarative Smart Contracts, and Blockchain Systems (Governatori et al.)	X	X	X	X
6 ACM	Efficient Publicly Verifiable 2PC over a Blockchain with Applications to Financially-Secure Computations (Zhu et al.)	X	X		
7 ACM	Robonomics: The Study of Robot-Human Peer-to-Peer Financial Transactions and Agreements (Cardenas and Kim)	X	X		
8 ACM	Blockchain-Oriented Software Engineering: Challenges and New Directions (Porru et al.)	X			
9 ACM	The Quest for Fully Smart Autonomous Business Networks in IoT Platforms (Ali et al.)	X	X		
10 ACM	Preliminary Steps towards Modeling Blockchain Oriented Software (Rocha and Ducasse)	X			
11 ACM	BitML: A Calculus for Bitcoin Smart Contracts (Bartoletti and Zunino)	X	X		
12 ACM	Understanding the Software Development Practices of Blockchain Projects: A Survey (Chakraborty et al.)	X	X		
13 ACM	A Method for Testing and Validating Executable Statechart Models (Mens et al.)	X	X		

Library	Title	Selected in step			Selected
		1	2	3	
14 ACM	Principles of Usable Programming Language Design (Coblenz)	X	X		
15 ACM	On the Implementation of Business Process Logic in DLT Nodes (Osterland et al.)	X			
16 ACM	Verifiable State Machines: Proofs That Untrusted Services Operate Correctly (Setty et al.)	X			
17 ACM	Trade-Offs between Distributed Ledger Technology Characteristics (Kannengieber et al.)	X			
18 ACM	Measurements, Analyses, and Insights on the Entire Ethereum Blockchain Network (Lee et al.)	X			
19 ACM	IContractML 2.0: A Domain-Specific Language for Modeling and Deploying Smart Contracts onto Multiple Blockchain Platforms (Hamdaqa et al.)	X	X		
20 ACM	Makina: A New QuickCheck State Machine Library (Barrio et al.)	X			
21 ACM	PLIERS: A Process That Integrates User-Centered Methods into Programming Language Design (Coblenz et al.)	X			
22 IEEE	Smart contracts vulnerabilities: a call for blockchain software engineering? (Destefanis et al.)	X	X		
23 IEEE	Formal Specification Technique in Smart Contract Verification (Lee et al.)	X	X		
24 IEEE	Auto-Generation of Smart Contracts from Domain-Specific Ontologies and Semantic Rules (Choudhury et al.)	X	X		
25 IEEE	Invited Paper: Beagle: A New Framework for Smart Contracts Taking Account of Law (Tsai et al.)	X	X	X	
26 IEEE	Blockchain-Oriented Software Engineering: Challenges and New Directions (Porru et al.)	X	X		

Library	Title	Selected in step			Selected
		1	2	3	
27 IEEE	Towards Model-Driven Engineering of Smart Contracts for Cyber-Physical Systems (Garamvölgyi et al.)	X	X	X	X
28 IEEE	Formal Verification of Smart Contracts Using Interface Automata (Madl et al.)	X	X	X	X
29 IEEE	Formal Requirement Enforcement on Smart Contracts Based on Linear Dynamic Logic (Sato et al.)	X	X		
30 IEEE	Towards Human-readable Smart Contracts (Franz et al.)	X	X		
31 IEEE	Design Patterns for Smart Contracts in the Ethereum Ecosystem (Wöhrrer and Zdun)	X	X		
32 IEEE	Preliminary Steps Towards Modeling Blockchain Oriented Software (Rocha and Ducasse)	X	X		
33 IEEE	Automatic smart contract generation using controlled natural language and template (Tateishi et al.)	X			
34 IEEE	Developing Safe Smart Contracts (Rezaei et al.)	X			
35 IEEE	Model-Based Software Design and Testing in Blockchain Smart Contracts: A Systematic Literature Review (Sánchez-Gómez et al.)	X	X		
36 IEEE	Verified Development and Deployment of Multiple Interacting Smart Contracts with VeriSolid (Nelaturu et al.)	X			
37 IEEE	Democratization of Smart Contracts: A Prototype for Automated Contract Generation (Franz et al.)	X			
38 IEEE	An ontological analysis of artifact-centric business processes managed by smart contracts (Van Wingerde and Weigand)	X			
39 IEEE	From Domain-Specific Language to Code: Smart Contracts and the Application of Design Patterns (Wohrer and Zdun)	X	X	X	

Library	Title	Selected in step			Selected
		1	2	3	
40 IEEE	Blockchain-based Supply Chain for the Automation of Transaction Process: Case Study based Validation (Habib et al.)	X			
41 IEEE	Towards Model checking approach for Smart contract validation in the EIP-1559 Ethereum (Fekih et al.)	X			
41 IEEE	The Notarial Office in E-government: A Blockchain-Based Solution (Gao et al.)	X			
42 IEEE	Solidity Code Generation From UML State Machines in Model-Driven Smart Contract Development (Jurgelaitis et al.)	X			
43 IEEE	From BPMN to smart contracts on blockchains: Transforming BPMN to DE-HSM multi-modal model (Liu et al.)	X			
44 IEEE	A Model for Verification and Validation of Law Compliance of Smart Contracts in IoT Environment (Amato et al.)	X	X	X	X
45 IEEE	A Survey on Blockchain Acquainted Software Requirements Engineering: Model, Opportunities, Challenges, and Future Directions (Farooq et al.)	X			
46 IEEE	Automatic Generation of Ethereum-Based Smart Contracts for Agri-Food Traceability System (Marchesi et al.)	X			
47 IEEE	Goal and Policy Based Code Generation and Deployment of Smart Contracts (Tsiounisa and Konstantinos)	X	X		
48 IEEE	Hierarchical Permissioned Blockchain and Traceability Of Requirement Changes (Rocky et al.)	X			
49 IEEE	A Smart Contract for Coffee Transport and Storage With Data Validation (Valencia-Payan et al.)	X			
50 IEEE	Model-driven approach for the design of Multi-Chain Smart Contracts (Barisic et al.)	X			

Library	Title	Selected in step			Selected
		1	2	3	
51 IEEE	MDE4BBIS: A Framework to Incorporate Model-Driven Engineering in the Development of Blockchain-Based Information Systems (de Souza and Burnay)	X	X		
52 IEEE	SPESC-Translator: Towards Automatically Smart Legal Contract Conversion for Blockchain-based Auction Services (Chen et al.)	X	X	X	X
53 Google Scholar	Symboleo: Towards a Specification Language for Legal Contracts (Sharif et al.)	X	X	X	X
54 Google Scholar	Formal Requirement Enforcement on Smart Contracts Based on Linear Dynamic Logic (Sato et al.)	X			
55 Google Scholar	Smart Contracts Using Blockly: Representing a Purchase Agreement Using a Graphical Programming Language (Weingaertner et al.)	X	X	X	X
56 Google Scholar	Model-Driven Engineering for Multi-party Interactions on a Blockchain - An Example (Dittmann et al.)	X	X		
57 Google Scholar	A Survey of Smart Contract Formal Specification and Verification (Tolmach et al.)	X	X		
58 Google Scholar	Design of the Blockchain Smart Contract: A Use Case for Real Estate (Karamitsos et al.)	X	X		
59 Google Scholar	Smart Contract Design Meets State Machine Synthesis: Case Studies (Suvorov and Ulyantsev)	X			
60 Google Scholar	Formal Verification of Functional Requirements for Smart Contract Compositions in Supply Chain Management Systems (Alqahtani et al.)	X	X	X	X

Library	Title	Selected in step			Selected
		1	2	3	
61 Google Scholar	A blockchain-based Decentralized System for proper handling of temporary Employment contracts (Pinna and Ibba)	X			
62 Google Scholar	From Domain-Specific Language to Code: Smart Contracts and the Application of Design Patterns (Wohrer and Zdun)	X			
63 Google Scholar	Literature Review: Smart Contract Semantics (Mathur)	X	X		
64 Google Scholar	Towards the Specification and Verification of Legal Contracts (Parvizimosaed)	X	X		
65 Google Scholar	The Extended UTXO Model (Chakravarty)	X	X	X	
66 Google Scholar	An ontological analysis of artifact-centric business processes managed by smart contracts (Van Wingerde and Weigand)	X			
67 Google Scholar	A Smart Contracting Framework for Aggregators of Demand-Side Response (Elizondo et al.)	X	X	X	X
68 Google Scholar	Modeling Business Processes and Contractual Agreements with Extended Finite State Machines (Flumini et al.)	X	X		
69 Google Scholar	Modeling and Analyzing Smart Contracts using Predicate Transition Nets (He)	X	X	X	X
70 Google Scholar	Generating and adjudicating digital legal agreements using Ethereum smart contracts (Liu et al.)	X	X		
71 Google Scholar	Enforcing commitments with blockchain: an approach to generate smart contracts for choreographed business processes (Bertolini)	X	X	X	X

Library	Title	Selected in step			Selected
		1	2	3	
72 Google Scholar	BDRM: A Blockchain-based Digital Rights Management Platform with Fine-grained Usage Control (Fei)	X	X		
73 Google Scholar	Blockchain Medicine Administration Records (BMAR): Reflections and Modelling Blockchain with UML. (Mitchell)	X	X	X	X
74 Google Scholar	An Architecture for Multi-chain Business Process Choreographies (Ladleif et al.)	X	X		
75 Google Scholar	Formally Verifying a Payment Channel System (Fu et al.)	X	X		
76 Google Scholar	Modeling Context-aware Legal Computing with Bigraphs (Yu et al.)	X	X		
77 Google Scholar	A Model-Driven Approach to Smart Contract Development (Boogard)	X	X	X	X
78 Google Scholar	An Ethereum-based Real Estate Application with Tampering-resilient Document Storage (Kopylash)	X			
79 Google Scholar	Blockchain Technology as a Regulatory Technology: From Code is Law to Law is Code (Filippi and Hassan)	X	X	X	
80 Google Scholar	B-MERODE: A Model-Driven Engineering and Artifact-Centric Approach to Generate Blockchain-Based Information Systems (Amaral de Sousa et al.)	X			
81 Google Scholar	CLoTH: a Simulator for HTLC Payment Networks (Conoscenti et al.)	X			
82 Google Scholar	Contract Design for Cloud Logistics (CL) Based on Blockchain Technology (BT) (Xu et al.)	X			

Library	Title	Selected in step			Selected
		1	2	3	
83 Google Scholar	Digital Certificates Using Blockchain: An Overview (Bhanushali et al.)	X			
84 Google Scholar	Housing Associations and Blockchain–A Positive Match? (Vonk et al.)	X			
85 Google Scholar	Mitigating software engineering costs in distributed ledger technologies (Heinecke et al.)	X			
86 Google Scholar	Model of Dynamic Smart Contract for Permissioned Blockchains. (Imeri et al.)	X			
87 Google Scholar	Smart Contract modeling and verification techniques: A survey (Imeri et al.)	X			
88 Google Scholar	Smart Contracts that are Smart and can function as Legal Contracts (Von Wendland)	X			
89 Google Scholar	Software Engineering for DApp Smart Contracts Managing Workers Contracts. (Lallai et al.)	X			
90 Google Scholar	Using Blockchain Technology to Manage Membership and Legal Contracts in a Distributed Data Market (Schlarb)	X			
91 Google Scholar	Reactive Synthesis of Smart Contract Control Flows (Finkbeiner et al.)	X			
92 Google Scholar	Protocol-based Smart Contract Generation (Falcao et al.)	X			
93 Google Scholar	Ethereum’s Smart Contracts Construction and Development using Model Driven Engineering Technologies: a Review (Hsain et al.)	X			

Library	Title	Selected in step			Selected
		1	2	3	
94 Google Scholar	Legally Enforceable Smart-Contract Languages (Dwivedi et al.)	X	X		
95 Google Scholar	Solidity Code Generation From UML State Machines in Model-Driven Smart Contract Development (Jurgelaitis et al.)	X			
96 Google Scholar	I Told You Tomorrow: Practical Time-Locked Secrets using Smart Contracts (Bacis et al.)	X			
97 Google Scholar	From BPMN to smart contracts on blockchains: Transforming BPMN to DE-HSM multi-modal model (Liu et al.)	X			
98 Google Scholar	An Automated Modeling Method and Visualization Implementation of Smart Contracts (Meng et al.)	X			
99 Google Scholar	Augmenting cryptocurrency in smart supply chain (Viriyasitavat et al.)	X			
100 Google Scholar	iContractML 2.0: A domain-specific language for modeling and deploying smart contracts onto multiple blockchain platforms (Hamdaqa et al.)	X			
101 Google Scholar	Reconfigurable Smart Contracts for Renewable Energy Exchange with Re-Use of Verification Rules (Gorski et al.)	X	X		
102 Google Scholar	A Tool for Moving Blockchain Computations Off-Chain (Liu et al.)	X			
103 Google Scholar	Ricardian Contracts for Industry 4.0 via the Arrowhead Contract Proxy (Palm et al.)	X			
104 Google Scholar	A Model for Verification and Validation of Law Compliance of Smart Contracts in IoT Environment (Amato et al.)	X			

Library	Title	Selected in step			Selected
		1	2	3	
105 Google Scholar	Contract as automaton: representing a simple financial agreement in computational form (Flood et al.)	X			
106 Google Scholar	Eunomia: Anonymous and Secure Vehicular Digital Forensics based on Blockchain (Li et al.)	X			
107 Google Scholar	Programming Legal Contracts (Crafa et al.)	X	X		
108 Google Scholar	Toward a Global Social Contract for Trade - a Rawlsian approach to Blockchain Systems Design and Responsible Trade Facilitation in the New Bretton Woods era (Lim et al.)	X			
109 Google Scholar	Model-Driven Development of Distributed Ledger Applications (Fraternali et al.)	X			
110 Google Scholar	Modelling the Development and Deployment of Decentralized Applications in Ethereum Blockchain: A BPMN-Based Approach (Nousias2022)	X			
111 Google Scholar	Blockchain software patterns for the design of decentralized applications: A systematic literature review (Six et al.)	X			
112 Google Scholar	A Hoare Logic with Regular Behavioral Specifications (Ernst et al.)	X			
113 Google Scholar	Overview of Test Coverage Criteria for Test Case Generation from Finite State Machines Modelled as Directed Graphs (Rechtberger et al.)	X			
114 Google Scholar	Blockchain Application Development Using Model-Driven Engineering and Low-Code Platforms: A Survey (Curty et al.)	X			

Library	Title	Selected in step			Selected
		1	2	3	
115 Google Scholar	A Blockchain Based Methodology for Power Grid Control Systems (Abdallah el al.)	X			
116 Google Scholar	The Effect of Thickness-Based Dynamic Matching Mechanism on a Hyperledger Fabric-Based TimeBank System (Lin el al.)	X			
117 Google Scholar	Formalizing the Blockchain-Based Block-Voke Protocol for Fast Certificate Revocation Using Colored Petri Nets (Sujatanagarjuna2021)	X			
118 Google Scholar	Cloud manufacturing service composition in IoT applications: a formal verification-based approach (Souri el al.)	X			
119 Google Scholar	Formalism-Driven Development: Concepts, Taxonomy, and Practice (Ding el al.)	X			
120 Google Scholar	An Agent-Oriented, Blockchain-Based Design of the Interbank Money Market Trading System (Alaeddini el al.)	X			
121 Google Scholar	Makina: a new QuickCheck state machine library (Barrio el al.)	X			
122 Google Scholar	Goal and Policy Based Code Generation and Deployment of Smart Contracts (Tsiounis el al.)	X			
123 Google Scholar	Chat2Code: Towards conversational concrete syntax for model specification and code generation, the case of smart contracts (Qasse el al.)	X			
124 Google Scholar	Automatic Generation of Ethereum-Based Smart Contracts for Agri-Food Traceability System (Marchesi el al.)	X			
125 Google Scholar	Secure MDE for Ethereum-based Decentralized Applications (DAppa) (Samreen el al.)	X	X		

Library	Title	Selected in step			Selected
		1	2	3	
126 Google Scholar	SPESC-Translator: Towards Automatically Smart Legal Contract Conversion for Blockchain-based Auction Services (Chen et al.)	X			
127 Google Scholar	PLIERS (Coblenz et al.)	X			
128 Google Scholar	Social Requirements Models for Services (Mylopoulos et al.)	X			
129 Google Scholar	Formalism- Driven Development of Decentralized Systems (Ding et al.)	X			
130 Google Scholar	Blockchain support for execution, monitoring and discovery of inter-organizational business processes (Sandoval et al.)	X			
131 Scopus	An overview on smart contracts: Challenges, advances and platforms (Zheng et al.)	X	X		
132 Scopus	Building Executable Secure Design Models for Smart Contracts with Formal Methods (Xu and Fink)	X	X	X	X
133 Scopus	Formal verification of workflow policies for smart contracts in azure blockchain (Wang et al.)	X	X	X	
134 Scopus	Verification-Led Smart Contracts (Banach)	X	X	X	X
135 Scopus	Das Contract - A Visual Domain Specific Language for Modeling Blockchain Smart Contracts (Skotnica and Pergl)	X	X	X	X
136 Scopus	Gigahorse: Thorough, Declarative Decompile of Smart Contracts (Grech et al.)	X	X		
137 Scopus	Building an executable axiomatisation of the REA2 ontology (Laurier and Horiuchi)	X	X		
138 Scopus	A unifying model of legal smart contracts (Ladleif and Weske)	X	X	X	

Library	Title	Selected in step			Selected
		1	2	3	
139 Scopus	VeriSolid: Correct-by-Design Smart Contracts for Ethereum (Mavridou et al.)	X	X	X	X
140 Scopus	Smart contracts as techno-legal regulation (Hunn)	X	X	X	
141 Scopus	Solidworx: A resilient and trustworthy transactive platform for smart and connected communities (Eisele et al.)	X	X		
142 Scopus	Formal modeling and verification of blockchain system (Duan et al.)	X	X		
143 Scopus	Tool Demonstration: FSolidM for designing secure ethereum smart contracts (Mavridou and Laszka)	X	X		
144 Scopus	Designing Secure Ethereum Smart Contracts: A Finite State Machine Based Approach (Mavridou and Laszka)	X	X	X	X
145 Scopus	Smart contracts and opportunities for formal methods (Miller et al.)	X	X		
146 Scopus	Inter-organizational Business Processes Managed by Blockchain (Nakamura et al.)	X	X		
147 Scopus	Towards human-readable smart contracts (Franz et al.)	X			
148 Scopus	Automatic smart contract generation using controlled natural language and template (Tateishi et al.)	X			
149 Scopus	BitML: A calculus for bitcoin smart contracts (Bartoletti and Zunino)	X			
150 Scopus	Towards Model-Driven Engineering of Smart Contracts for Cyber-Physical Systems (Garamvölgyi et al.)	X			
151 Scopus	Design Patterns for Smart Contracts in the Ethereum Ecosystem (Wohrer and Zdun)	X			
152 Scopus	Formal modeling and verification of smart contracts (Bai et al.)	X			
153 Scopus	Applications of model-driven engineering in cyber-physical systems: A systematic mapping study (Mohamed et al.)	X	X		

Library	Title	Selected in step			Selected
		1	2	3	
154 Scopus	Democratization of Smart Contracts: A Prototype for Automated Contract Generation (Franz et al.)	X	X	X	
155 Scopus	Domain Specific Language for Smart Contract Development (Wohrer and Zdun)	X	X	X	
156 Scopus	Smart Contracts for Government Processes: Case Study and Prototype Implementation (Short Paper) (Krogsbøll et al.)	X	X		
157 Scopus	Towards an automated DEMO action model implementation using blockchain smart contracts (Aparício et al.)	X	X		
158 Scopus	An Automated Modeling Method and Visualization Implementation of Smart Contracts (Meng et al.)	X	X		
159 Scopus	Automatic Generation of Ethereum-Based Smart Contracts for Agri-Food Traceability System (Marchesi et al.)	X			
160 Scopus	Automating Smart Contract Generation on Blockchains Using Multi-modal Modeling (Liu et al.)	X			
161 Scopus	Contract as automaton: representing a simple financial agreement in computational form (Flood et al.)	X	X	X	X
162 Scopus	Design and development of smart contracts for E-government through value and business process modeling (Gomez et al.)	X	X	X	
163 Scopus	Ethereum's smart contracts construction and development using model driven engineering technologies: A review (AitHsain et al.)	X	X		
164 Scopus	From BPMN to smart contracts on blockchains: Transforming BPMN to DE-HSM multi-modal model (Liu et al.)	X	X		
165 Scopus	From the Graphical Representation to the Smart Contract Language: A Use Case in the Construction Industry (Ye et al.)	X	X	X	

Library	Title	Selected in step			Selected
		1	2	3	
166 Scopus	iContractML 2.0: A domain-specific language for modeling and deploying smart contracts onto multiple blockchain platforms (Hamdaqa et al.)	X			
167 Scopus	Model-driven approach for the design of Multi-Chain Smart Contracts (Barisic et al.)	X	X		
168 Scopus	Overview of Test Coverage Criteria for Test Case Generation from Finite State Machines Modelled as Directed Graphs (Rechtberger et al.)	X			
169 Scopus	Solidity Code Generation From UML State Machines in Model-Driven Smart Contract Development (Jurgelaitis et al.)	X	X	X	X
170 Springer	On Refining Design Patterns for Smart Contracts (Zecchini et al.)	X	X		
171 Springer	Verifying Smart Contracts with Cubicle (Conchon et al.)	X	X	X	X
172 Springer	Blockchain-based intelligent contract for factoring business in supply chains (Zheng et al.)	X	X		
173 Springer	Analysis of Ethereum Smart Contracts and Opcodes (Bistarelli et al.)	X	X		
174 Springer	Systems and Methods for Implementing Deterministic Finite Automata (DFA) via a Blockchain (Wright)	X	X		
175 Springer	Ghazal: Toward Truly Authoritative Web Certificates Using Ethereum (Moosavi and Clark)	X	X		
176 Springer	A Legal Interpretation of Choreography Models (Ladleif and Weske)	X	X	X	X
177 Springer	Modeling and execution of blockchain-aware business processes (Falazi et al.)	X	X		
178 Springer	A Blockchain-Based Decentralized System for Proper Handling of Temporary Employment Contracts (Pinna and Ibba)	X	X	X	X

Library	Title	Selected in step			Selected
		1	2	3	
179 Springer	On legal contracts, imperative and declarative smart contracts, and blockchain systems (Governatori et al.)	X	X	X	
180 Springer	A Semantic Framework for the Security Analysis of Ethereum Smart Contracts (Grishchenko et al.)	X	X		
181 Springer	Empowering Business-Level Blockchain Users with a Rules Framework for Smart Contracts (Astigarraga et al.)	X	X		
182 Springer	SoK: Unraveling Bitcoin Smart Contracts (Atzei et al.)	X	X		
183 Springer	Automated Execution of Financial Contracts on Blockchains (Egelund-Müller et al.)	X	X		
184 Springer	Towards a Shared Ledger Business Collaboration Language Based on Data-Aware Processes (Hull et al.)	X	X		
185 Springer	Building Executable Secure Design Models for Smart Contracts with Formal Methods (Xu and Fink.)	X			
186 Springer	Verification-Led Smart Contracts (Banach)	X			
187 Springer	Smart Contracts and Opportunities for Formal Methods (Miller et al.)	X			
188 Springer	Tool Demonstration: FSolidM for Designing Secure Ethereum Smart Contracts (Mavridou and Laszka)	X			
189 Springer	A method for testing and validating executable statechart models (Mens et al.)	X			
190 Springer	A Unifying Model of Legal Smart Contracts (Ladleif and Weske)	X	X	X	
191 Springer	Designing Secure Ethereum Smart Contracts: A Finite State Machine Based Approach (Mavridou and Laszka)	X			

Library	Title	Selected in step			Selected
		1	2	3	
192 Springer	Inter-organizational Business Processes Managed by Blockchain (Nakamura et al.)	X			
193 Springer	Modeling and reasoning about uncertainty in goal models: a decision-theoretic approach (Liaskos et al.)	X			
194 Springer	Comprehensive review of modeling, structure, and integration techniques of smart buildings in the cyber-physical-social system (Gong et al.)	X			
195 Springer	Engineering Multi-agent Systems with Statecharts (Spanoudakis et al.)	X	X		
196 Springer	A Blockchain-Based System for Agri-Food Supply Chain Traceability Management (Marchese et al.)	X			
197 Springer	Business Process Engineering for Data Storing and Processing in a Collaborative Distributed Environment Based on Provenance Metadata, Smart Contracts (Demichev et al.) and Blockchain Technology	X			
198 Springer	Blockchain Medicine Administration Records (BMAR): Reflections and Modelling Blockchain with UML (Mitchell et al.)	X	X		
199 Springer	Model-Driven Development of Distributed Ledger Applications (Fraternali et al.)	X			
200 Springer	Blockchain Application Development Using Model-Driven Engineering and Low-Code Platforms: A Survey (Curty et al.)	X			
201 Springer	Modelling the Development and Deployment of Decentralized Applications in Ethereum Blockchain: A BPMN-Based Approach (Nousias et al.)	X			

Library	Title	Selected in step			Selected
		1	2	3	
202 Springer	An Agent-Oriented, Blockchain-Based Design of the Interbank Money Market Trading System (Alaeddini et al.)	X			
203 Springer	Automated Consistency Analysis for Legal Contracts (Khoja et al.)	X			

APPENDIX B – JSON GENERATED WITH THE ALGORITHM

```
1  {
2    "name": "serviceSaleC",
3    "states": {
4      "created": "Created",
5      "in_effect": "In Effect",
6      "suspended": "Suspended",
7      "successful_termination": "Successful Termination",
8      "unsuccessful_termination": "Unsuccessful Termination"
9    },
10   "transitions": {
11     "create_contract": {
12       "source": "initial",
13       "target": "created",
14       "events": [
15         {
16           "event": "Create",
17           "guard": "",
18           "actions": [
19             "Assign parties ${parties}",
20             "Create obligations ${obligations}",
21             "Create powers ${powers}"
22           ]
23         }
24       ],
25       "parties": [
26         "techCompanySeller",
27         "techCompanyBuyer"
28       ],
29       "obligations": [
30         "01.5",
31         "02.5III"
32       ],
33       "powers": []
34     },
35
```

Figure 41 – Final json generated by algorithm part 1

```
1  "activate_contract": {
2    "source": "created",
3    "target": "in_effect",
4    "events": [
5      {
6        "event": "Activate",
7        "guard": "effectiveDate = actualDate",
8        "actions": [
9          "Activate obligations ${obligations}",
10         "Activate powers ${powers}"
11       ]
12     }
13   ],
14   "obligations": [
15     "01.5",
16     "02.5III"
17   ],
18   "powers": []
19 },
20 "activate_obligation_power": {
21   "source": "in_effect",
22   "target": "in_effect",
23   "events": [
24     {
25       "event": "Obligation violation <strong>02.5III</strong>",
26       "guard": "",
27       "actions": [
28         "Activate obligation <strong>03.5</strong>"
29       ]
30     },
31     {
32       "event": "happens <strong>endOfMonth</strong>",
33       "guard": "",
34       "actions": [
35         "Activate obligation <strong>02.3VII</strong>"
36       ]
37     },
38     {
39       "event": "Obligation violation <strong>02.3VII</strong>",
40       "guard": "",
41       "actions": [
42         "Activate power <strong>P3.5</strong>"
43       ]
44     },
45     {
46       "event": "happens <strong>rescissionDeal</strong>",
47       "guard": "",
48       "actions": [
49         "Activate power <strong>P12.5</strong>"
50       ]
51     }
52   ]
53 },
54
```

Figure 42 – Final json generated by algorithm part 2


```
1  "replace_party": {
2    "source": "Active",
3    "target": "Active",
4    "events": [
5      {
6        "event": "Replace party ${powers}",
7        "guard": "",
8        "actions": [
9          "Replace (${old_party}, ${new_party})"
10       ]
11     }
12   ],
13   "powers": [],
14   "old_party": "",
15   "new_party": ""
16 },
17 "fulfill_active_obligations": {
18   "source": "in_effect",
19   "target": "successful_termination",
20   "events": [
21     {
22       "event": "Fulfill active obligations ${set_of_obligations}",
23       "guard": "",
24       "actions": []
25     }
26   ],
27   "set_of_obligations": [
28     [
29       "01.5",
30       "02.5III",
31       "02.3VII"
32     ],
33     [
34       "01.5",
35       "03.5",
36       "02.3VII"
37     ]
38   ]
39 },
40 "activate_surviving_obligation": {
41   "source": "successful_termination",
42   "target": "successful_termination",
43   "events": [
44     {
45       "event": "happens <strong>MaintenanceByDemandOption</strong>",
46       "guard": "",
47       "actions": [
48         "Activate obligation <strong>S04.2</strong>"
49       ]
50     },
51     {
52       "event": "happens <strong>maintenanceByMonthOption</strong>",
53       "guard": "",
54       "actions": [
55         "Activate obligation <strong>S04.3</strong>"
56       ]
57     }
58   ]
59 },
60
```

Figure 43 – Final json generated by algorithm part 3

```
1  "terminate_contract": {
2    "source": "Active",
3    "target": "unsuccessful_termination",
4    "event": "terminate",
5    "events": [
6      {
7        "event": "Terminate [ ${powers} ]",
8        "guard": "",
9        "actions": []
10     },
11     {
12       "event": "Unfulfilled obligations ${obligations_unfulfilled}",
13       "guard": "",
14       "actions": []
15     }
16   ],
17   "powers": [
18     "P12.5"
19   ],
20   "obligations_unfulfilled": [
21     [
22       "~ 01.5",
23       "~ 02.5III",
24       "~ 02.3VII"
25     ],
26     [
27       "~ 01.5",
28       "~ 03.5",
29       "~ 02.3VII"
30     ]
31   ]
32 }
33 },
34 "state_actions": {
35   "successful_termination": {
36     "when": "+ On Enter",
37     "action": "Activate surviving obligations ${surviving_obligations}",
38     "surviving_obligations": [
39       "S06.1"
40     ]
41   },
42   "unsuccessful_termination": {
43     "when": "+ On Enter",
44     "action": "Activate surviving obligations ${surviving_obligations}",
45     "surviving_obligations": [
46       "S05.2"
47     ]
48   }
49 }
50 ;
```

Figure 44 – Final json generated by algorithm part 4

ANNEX A - CONTRACT FOR EXPERIMENT 1

**SERVIÇO PÚBLICO FEDERAL
UNIVERSIDADE FEDERAL DE SANTA CATARINA****Pró-Reitoria de Administração - PROAD
Departamento de Projetos, Contratos e Convênios - DPC**

Avenida Desembargador Vitor Lima, nº 222, 8º andar (Sala 802), Prédio da Reitoria 2
Bairro Trindade – Florianópolis/SC – CEP 88.040-400

CNPJ/MF nº 83.899.526/0001-82

Telefones: (48) 3721-4234/3721-4240/3721-4236

E-mail: dpc.proad@contato.ufsc.br

**TERMO DE CONTRATO QUE CELEBRAM ENTRE SI A
UNIVERSIDADE FEDERAL DE SANTA CATARINA E A
EMPRESA DICAPEL PAPÉIS E EMBALAGENS LTDA.**

A Universidade Federal de Santa Catarina (UFSC), autarquia educacional criada e integrada ao Ministério da Educação (MEC) pela Lei n.º 3.849, de 18/12/1960, inscrita no CNPJ/MF sob o n.º 83.899.526/0001-82, com sede no Campus Universitário, Bairro Trindade, nesta Capital, representada pela Pró-Reitora de Administração em exercício, Daiana Prigol Bonetti, CPF n.º 064.945.129-50, doravante denominada CONTRATANTE e a Empresa **DICAPEL PAPÉIS E EMBALAGENS LTDA**, inscrita no CNPJ n.º 83.413.591/0003-18, com sede na Rua Dois de Setembro, nº 305, Sala B, bairro Itoupava Norte, em Blumenau/SC, CEP 89.052-000, doravante denominada CONTRATADA, neste ato representada pelo Sr. Edson Fernando Mazzuco, CPF n.º 023.627.449-06, firmam o presente TERMO de contrato, de acordo com o **Processo n.º 23080.036950/2018-62 e Solicitação Digital n.º 049848/2019**, com sujeição às normas emanadas da Lei n.º 8.666/93 e suas alterações posteriores, Lei n.º 10.520/02 e suas alterações posteriores, Lei n.º 9.784/99 e suas alterações, Decreto n.º 5.450/05 e suas alterações, Decreto n.º 7.892/13 e suas alterações, e às disposições estabelecidas no **Edital de Pregão n.º 243/2018** e nas complementações a ele integradas, aos termos da proposta vencedora e sob as seguintes cláusulas e condições:

CLÁUSULA PRIMEIRA – OBJETO

- 1.1. Este contrato tem como objeto a **aquisição de materiais gráficos para a Imprensa Universitária, da Universidade Federal de Santa Catarina - UFSC**, conforme quantidades, valores unitários e totais estabelecidos neste instrumento, conforme ANEXO I.
- 1.2. Este Termo de Contrato vincula-se ao Edital do Pregão identificado no preâmbulo e à proposta vencedora, independentemente de transcrição.

CLÁUSULA SEGUNDA - DO LOCAL E DO PRAZO DE ENTREGA DO OBJETO

- 2.1. Os produtos/materiais deverão ser entregues na **Universidade Federal de Santa Catarina, no Campus Universitário Reitor João David Ferreira Lima, Bairro Trindade, em Florianópolis/SC, CEP 88040-900**, ou em outro local definido na solicitação de fornecimento, em horário comercial, de segunda a sexta-feira das 08h00 às 12h00 e das 14h00 às 17h00.
 - 2.1.1. A Contratada deverá entrar em contato com o responsável pelo recebimento indicado na solicitação de fornecimento para **programar a entrega**.
- 2.2. O prazo para entrega dos itens que compõem o objeto desta licitação é de **10 (dez) dias (úteis)**, contados do recebimento da solicitação pelo fornecedor.

2.3. O encaminhamento da solicitação de fornecimento poderá ser efetuado mediante o envio, pela Administração, de correspondência eletrônica (e-mail) ao correio eletrônico da Contratada **constante do Sistema de Cadastramento Unificado de Fornecedores - SICAF ou na Ata de Registro de Preços assinada pela Contratada**. A confirmação do envio da solicitação será aferida mediante o recebimento de relatório de confirmação de entrega, a ser automaticamente encaminhado pelo sistema administrador de e-mails da UFSC, independentemente do envio de confirmação de leitura e/ou recebimento por parte da Contratada.

2.4. Eventuais pedidos de prorrogação de prazo de fornecimento deverão ser encaminhados, via ofício, para o endereço eletrônico **imprensa@contato.ufsc.br**, sendo obrigatória a menção ao item e ao Pregão a que se refere o pedido.

2.5. No caso de produtos/materiais perecíveis, o prazo de validade na data da entrega **não poderá ser inferior a 6 (seis) meses**, ou a **metade do prazo total** recomendado pelo fabricante.

CLÁUSULA TERCEIRA – DA VIGÊNCIA

3.1. O prazo de vigência do contrato será de **12 (doze) meses**, a partir da data da assinatura do instrumento, nos termos do artigo 57 da Lei nº 8.666/93, **sem possibilidade de prorrogação**.

3.1.1. As obrigações pertinentes à garantia contratual do objeto, previstas na cláusula sexta, têm prazo de vigência próprio e desvinculado do prazo acima citado, permitindo eventual aplicação de penalidades em caso de descumprimento de alguma de suas condições, mesmo depois de expirada a vigência contratual.

CLÁUSULA QUARTA - OBRIGAÇÕES DA CONTRATANTE

4.1. Acompanhar e fiscalizar a execução do fornecimento contratado, bem como realizar testes nos bens fornecidos, atestar nas notas fiscais/fatura a efetiva entrega do objeto contratado e o seu aceite.

4.2. Aplicar à Contratada as sanções regulamentares e contratuais.

4.3. Prestar as informações e os esclarecimentos solicitados pela Contratada, pertinentes ao objeto, para a fiel execução do avençado.

4.4. Informar à Contratada, toda e qualquer irregularidade constatada na execução do objeto.

CLÁUSULA QUINTA - DAS OBRIGAÇÕES DA CONTRATADA

5.1. Fornecer à Contratante a quantidade dos produtos/materiais discriminada na respectiva Autorização de Fornecimento, no prazo estabelecido no item 2.2 do presente Termo de Contrato.

5.2. Corrigir, a suas expensas, quaisquer danos causados à Contratante e/ou a terceiros.

5.3. Atender prontamente às exigências da Contratante inerentes ao objeto do fornecimento.

5.4. Manter, durante a execução do fornecimento contratado, as mesmas condições da habilitação.

5.5. Colocar à disposição da Contratante todos os meios necessários para comprovação da qualidade dos materiais, permitindo a verificação de sua conformidade com as especificações e exigências do Edital.

5.6. Comunicar à Contratante, no prazo máximo de 48 horas que antecede a data da entrega, os motivos que impossibilitem o cumprimento do prazo previsto, com a devida comprovação.

5.7. Responsabilizar-se pelas despesas dos tributos, encargos trabalhistas, previdenciários, fiscais, comerciais, tarifas, fretes, seguros, deslocamento de pessoal, prestação de garantia e quaisquer outras que incidam ou venham a incidir na execução do contrato.

5.8. Responsabilizar-se para que os bens sejam, preferencialmente, acondicionados em embalagem individual adequada, com o menor volume possível, que utilize materiais recicláveis, de forma a garantir a máxima proteção durante o transporte e o armazenamento.

5.9. Responder por qualquer prejuízo que seus empregados ou prepostos causarem ao patrimônio da Contratante e/ou a terceiros, decorrentes de ação ou omissão culposa ou dolosa, procedendo imediatamente aos reparos ou indenizações cabíveis e assumindo o ônus referente.

5.10. Não transferir a terceiros, por qualquer forma, nem mesmo parcialmente, as obrigações assumidas, nem subcontratar qualquer das prestações a que está obrigada.

5.11. Responsabilizar-se pelo transporte, acondicionamento e entrega, inclusive descarregamento dos materiais.

5.12. Assegurar-se de que os bens não contenham substâncias perigosas em concentração acima da recomendada na diretiva RoHS (Restriction of Certain Hazardous Substances), tais como mercúrio (Hg), chumbo (Pb), cromo hexavalente (Cr(VI)), cádmio (Cd), bifenil-polibromados (PBBs), éteres difenil-polibromados (PBDEs).

5.13. Responsabilizar-se pela retirada dos resíduos das embalagens do local de entrega e comprometer-se pela destinação correta dos mesmos.

CLÁUSULA SEXTA – GARANTIA

6.1. O prazo de garantia mínima será de **03 (três) meses ou a fornecida pelo fabricante, a que for maior**. Não havendo indicação expressa, será considerado como tal.

6.2. A garantia do produto, no prazo mínimo estipulado no item 6.1 deste Termo de Contrato, consiste na prestação, pela Contratada, de todas as obrigações previstas na Lei nº 8.078/1990 e suas posteriores alterações – Código de Defesa do Consumidor, bem como dos encargos previstos à Contratada no Edital e seus Anexos.

6.3. O aceite/aprovação do(s) produto(s)/material(is) pelo órgão licitante não exclui a responsabilidade civil do fornecedor por vícios de quantidade ou qualidade do(s) produto(s) ou disparidades com as especificações estabelecidas, verificadas, posteriormente, garantindo-se a UFSC as faculdades previstas no art. 18 da Lei nº 8.078/1990.

6.4. Caso, por qualquer razão, não possa ser processado o recebimento definitivo no momento da entrega, o objeto licitado será recebido provisoriamente para posterior verificação de sua conformidade com as especificações constantes da Nota de Empenho e do respectivo documento fiscal.

6.5. O produto/material que for entregue fora das condições estipuladas no Edital não será aceito, devendo ser substituído no prazo de até **5 (cinco) dias (úteis)**, sendo o ônus decorrente da substituição de responsabilidade da Contratada.

CLÁUSULA SÉTIMA – DO PAGAMENTO

7.1. O valor estimado a ser pago à CONTRATADA pelo objeto do presente contrato é de **RS 47.650,00 (quarenta e sete mil seiscentos e cinquenta reais)**.

7.2. A CONTRATANTE realizará o pagamento em conformidade com as condições previstas no Edital.

7.3. Os recursos necessários ao atendimento das despesas do presente contrato correrão à conta do Orçamento Geral da CONTRATANTE, no Programa de Trabalho: 12364208020RK0042 e 12368208020RI0042 ; PTRES 108366 e 108369; Natureza de Despesa 33903041; Fonte: 8100000000.

7.4. Os pagamentos na CONTRATANTE são realizados em conformidade com a Lei nº 8.666/1993 e conforme disponibilidade de recursos financeiros, pelo Departamento de Contabilidade e Finanças (DCF), mediante crédito bancário, salvo:

7.4.1 Os pagamentos decorrentes de despesas cujos valores não ultrapassem o limite de que trata o inciso II do art. 24, da Lei nº 8.666/1993, serão efetuados no prazo de até 5 (cinco) dias úteis, contados da apresentação da nota fiscal/fatura.

7.5. O pagamento será efetuado pelo DCF no prazo máximo de 30 (trinta) dias, a contar do recebimento dos materiais/prestação dos serviços e, assim como, da entrega da nota fiscal/fatura devidamente atestada, a qual deverá:

7.5.1. Ser emitida conforme as previsões legais e regulamentares vigentes, em 2 (duas) vias ou mais, com mesma razão social e número de inscrição no CNPJ/MF informados para a habilitação e oferecimento da proposta de preços, bem como deverá conter todos os dados necessários à perfeita compreensão do documento.

7.5.2. Conter registro da data de sua apresentação/recebimento e do servidor responsável por este em todas as suas vias, assim como, em mecanismo complementar de registro, como livro protocolo de recebimento, aviso de recebimento ou outro, quando houver.

7.6. Quando da ocorrência de eventuais atrasos de pagamento provocados exclusivamente pela Administração, o valor devido deverá ser acrescido de atualização financeira, e sua apuração se fará desde a data de seu vencimento até a data do efetivo pagamento, em que os juros de mora serão calculados à taxa de 0,5% (meio por cento) ao mês, ou 6% (seis por cento) ao ano, mediante aplicação das seguintes formulas:

$$I=(TX/100)$$

365

EM = I x N x VP, onde:

I = Índice de atualização financeira;

TX = Percentual da taxa de juros de mora anual;

EM = Encargos moratórios;

N = Número de dias entre a data prevista para o pagamento e a do efetivo pagamento;

VP = Valor da parcela em atraso.

7.6.1. Na hipótese de pagamento de juros de mora e demais encargos por atraso, os autos devem ser instruídos com as justificativas e motivos, e ser submetidos à apreciação da autoridade superior competente, que adotará as providências para verificar se é ou não caso de apuração de responsabilidade, identificação dos envolvidos e imputação de ônus a quem deu causa.

7.7. Será considerado como data do pagamento o dia em que constar como emitida a ordem bancária (OB) para pagamento.

7.8. O pagamento somente será autorizado depois de efetuado o “ateste” pelo servidor competente, devidamente identificado, na nota fiscal apresentada e depois de verificada a regularidade fiscal do prestador dos serviços.

7.9. Quando do pagamento, será efetuada a retenção tributária, nos termos da legislação aplicável.

7.9.1. Quanto ao Imposto sobre Serviços de Qualquer Natureza (ISSQN), será observado o disposto na Lei Complementar nº 116, de 2003, e legislação municipal aplicável.

7.9.2. A CONTRATADA regularmente optante pelo Simples Nacional, nos termos da Lei Complementar nº 123, de 2006, não sofrerá a retenção tributária quanto aos impostos e contribuições abrangidos por aquele regime. No entanto, o pagamento ficará condicionado à

apresentação de comprovação por meio de documento oficial de que faz jus ao tratamento tributário favorecido previsto na referida Lei Complementar.

7.10. A Administração deduzirá do montante a ser pago os valores correspondentes às multas e/ou indenizações devidas por parte da CONTRATADA.

7.10.1. O desconto de qualquer valor no pagamento devido a CONTRATADA será precedido de processo administrativo em que será garantido o contraditório e a ampla defesa, com os recursos e meios que lhes são inerentes.

7.11. É vedado a CONTRATADA transferir a terceiros os direitos ou créditos decorrentes do contrato.

7.12. Nenhum pagamento será efetuado ao fornecedor enquanto estiver pendente de liquidação qualquer obrigação financeira que lhe tiver sido imposta em decorrência de inadimplência contratual.

7.13. No interesse da Administração poderá ocorrer a antecipação de pagamento, sendo este em duas hipóteses:

7.13.1. Por meio de correspondência com a antecipação da execução da obrigação, propiciando descontos para a CONTRATADA (artigo 40, XIV, 'd'). Calculado à taxa de 0,5% (meio por cento) ao mês, ou 6% (seis por cento) ao ano, mediante aplicação da seguinte fórmula:

$$I = \frac{(TX/100)}{365}$$

$$D = I \times N \times VP, \text{ onde:}$$

I = Índice de atualização financeira;

TX = Percentual da taxa de desconto;

D = Desconto por antecipação;

N = Número de dias entre a data prevista para o pagamento e a do efetivo pagamento antecipado;

VP = Valor da parcela a ser antecipada.

7.13.2. Nas contratações internacionais, onde poderá prevalecer disposição especial a ser acordada entre as partes;

CLÁUSULA OITAVA – DO REAJUSTE/REVISÃO DE PREÇOS

8.1. Os preços são fixos e irredutíveis, exceto nos casos previstos no Decreto nº 7.892/13:

8.1.1. Os preços registrados poderão ser revistos em decorrência de eventual redução dos preços praticados no mercado ou de fato que eleve o custo dos serviços ou bens registrados, cabendo ao órgão gerenciador promover as negociações junto aos fornecedores, observadas as disposições contidas na alínea "d" do inciso II do caput do art. 65 da Lei nº 8.666/93;

8.1.2. Quando o preço registrado tornar-se superior ao preço praticado no mercado por motivo superveniente, o órgão gerenciador convocará os fornecedores para negociarem a redução dos preços aos valores praticados pelo mercado.

CLÁUSULA NONA – FISCALIZAÇÃO E ACOMPANHAMENTO

9.1. A CONTRATANTE designará um fiscal para acompanhar e controlar a execução do contrato, a qual será realizada em total observância ao contido no Edital e, ainda, aos regramentos legais da Lei nº 8.666/93.

CLÁUSULA DÉCIMA – SANÇÕES

10.1. O licitante/fornecedor que, convocado dentro do prazo de validade de sua proposta, não assinar a Ata de Registro de Preços, deixar de entregar documentação exigida neste Edital, apresentar documentação falsa, ensejar o retardamento da execução de seu objeto, não mantiver a proposta/lance, falhar ou fraudar na execução da Ata/Contrato, comportar-se de modo inidôneo, fizer declaração falsa ou cometer fraude fiscal, garantido o direito à ampla defesa, ficará impedido de licitar e de contratar com a União e será descredenciado do SICAF, pelo prazo de até 5 (cinco) anos, sem prejuízo das sanções previstas neste Edital e na Ata de Registro de Preços e das demais cominações legais.

10.2. Pela inexecução total ou parcial do contrato a Administração poderá, garantida a prévia defesa, aplicar à CONTRATADA as sanções previstas no item 18 do **Edital do Pregão n.º243/2018** desta instituição, com seus Anexos.

CLÁUSULA DÉCIMA PRIMEIRA- DA ALTERAÇÃO E RESCISÃO

11.1. A alteração deste contrato poderá ocorrer em consonância com o art.12 do Decreto nº 7.892/13:

11.1.1. Os contratos decorrentes do Sistema de Registro de Preços poderão ser alterados, observado o disposto no art. 65 da Lei n.º 8.666/93;

11.1.2. Em caso de alteração contratual, o mesmo será formalizado por meio de termo aditivo, a ser assinado pelas partes;

11.1.3. Atos que não caracterizem alteração de contrato poderão ser registrados por simples apostilamento, dispensando a celebração de aditamento.

11.2. A rescisão deste contrato poderá ser:

11.2.1. Determinada por ato unilateral e escrito da CONTRATANTE, nos casos enumerados nos incisos I a XII do artigo 78 da Lei nº 8.666/93, notificando-se a CONTRATADA com antecedência mínima de 30 (trinta) dias;

11.2.2. Amigável, por acordo entre as partes, reduzido a termo, desde que haja conveniência para a CONTRATANTE;

11.2.3. Judicial, nos termos da legislação vigente sobre a matéria;

11.2.4. Determinado por ato unilateral e escrito da CONTRATANTE, nos casos enumerados nos artigos 19,20 e 21 do Decreto nº 7.892/13.

11.3. A CONTRATADA reconhece os direitos da CONTRATANTE em caso de rescisão administrativa prevista no art. 77 da Lei nº 8.666, de 1993. O termo de rescisão, sempre que possível, será precedido:

11.3.1. Balanço dos eventos contratuais já cumpridos ou parcialmente cumpridos;

11.3.2. Relação dos pagamentos já efetuados e ainda devidos;

11.3.3. Indenizações e multas.

CLÁUSULA DÉCIMA SEGUNDA– DAS DISPOSIÇÕES GERAIS

12.1. É vedado efetuar acréscimos nos quantitativos fixados pela ata de registro de preços que deu causa e motivação a este termo de contrato, inclusive o acréscimo de que trata o § 1º do art. 65 da Lei nº 8Lei nº 8.666/93.

12.2. As questões e os litígios oriundos do presente contrato e não dirimidos consensualmente serão resolvidos na Subseção Judiciária de Florianópolis (Seção Judiciária de Santa Catarina) - Justiça Federal.

12.3. Integra este Termo de Contrato o Anexo I, contemplado na página nº 8, que detalha a lista dos produtos/materiais e quantitativos adquiridos pela CONTRATADA.

12.4. A publicação resumida do instrumento de contrato ou de seus aditamentos na Imprensa Oficial, que é condição indispensável para sua eficácia, será providenciada pela Administração.

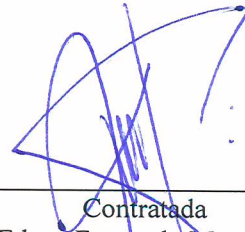
E, por estarem assim justas e acordadas, firmam as partes o presente instrumento em três vias de igual teor e forma, na presença das testemunhas abaixo.

Florianópolis, 24 de julho de 2019.



Contratante
Daiana Prigol Bonetti
CPF: 064.945.129-50

Daiana Prigol Bonetti
CPF: 064.945.129-50
Poderão nº 1471/2019/0R

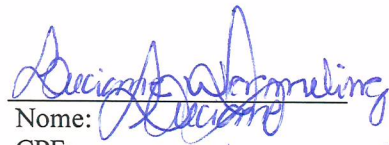


Contratada
Edson Fernando Mazzuco
CPF: 023.627.449-06

Testemunhas:



Nome: ULISSES IRACI ZILIO
CPF: 004.595.099-77



Nome: LUCIANA DE AGUIAR
CPF: 896.290.099-49

ANEXO I

Grupo/ Item	Descrição	Unid. Medida	Qtde.	Valor Unitário (R\$)	Valor Total (R\$)
0002	275612 - PAPEL COUCHE LISO 150 G - FOLHA PACOTE COM 250 FOLHAS PAPEL COUCHÊ LISO, 150G/M2, FORMATO 960 MM X 660 MM.	FL	50.000	0,525	26.250,00
0006	280161 - PAPEL OFFSET 90 G - FOLHA PACOTE COM 250 FOLHAS. PAPEL OFFSET, 90G/M2, BRANCO, FORMATO 960 MM X 660 MM.	FL	20.000	0,38	7.600,00
0009	238769 - PAPEL OFFSET 180 G - FOLHA PACOTE COM 125 FOLHAS. PAPEL OFFSET, 180G/M2, BRANCO, FORMATO 960 MM X 660 MM.	FL	20.000	0,69	13.800,00
Total					47.650,00





SERVIÇO PÚBLICO FEDERAL
UNIVERSIDADE FEDERAL DE SANTA CATARINA
PRÓ-REITORIA DE ADMINISTRAÇÃO
DEPARTAMENTO DE PROJETOS, CONTRATOS E CONVÊNIOS
Campus Universitário Reitor João David Ferreira Lima - Trindade
CEP: 88040-900 - Florianópolis - SC

PORTARIA Nº 0200/2019/DPC DE 30 de Julho de 2019.

O(A) Diretor(a) do Departamento de Projetos, Contratos e Convênios, no uso de suas atribuições, delegadas pela Portaria nº 295/PROAD/2017, de 02 de agosto de 2017 e de acordo com o previsto no Art. 67 da Lei nº 8.666, de 21 de junho de 1993 e legislação correlata, **RESOLVE:**

Art. 1º - Designar, conforme disciplinado na Instrução Normativa nº 05/SLTI/MP, de 26 de maio de 2017, os servidores/setores abaixo relacionados, para gerenciar, acompanhar e fiscalizar a execução e o adequado cumprimento das cláusulas estabelecidas no Contrato nº 00144/2019 (processo 036950/2018-62), celebrado entre a UNIVERSIDADE FEDERAL DE SANTA CATARINA e a empresa/instituição DICAPEL PAPÉIS E EMBALAGENS LTDA., CNPJ nº 83.413.591/0003-18.

FUNÇÃO	NOME	CPF
Fiscal Administrativo	MAURO JOSE ELIAS	564859489-53
Fiscal Administrativo	CESAR MURILO NATIVIDADE	485926689-72

Art. 2º - Esta Portaria entra em vigor na data de sua publicação e terá vigência até o vencimento do contrato e de sua garantia quando houver.

Diretor do Departamento de Projetos, Contratos
e Convênios

ANNEX B - CONTRACT FOR EXPERIMENT 2

CONTRATO DE PRESTAÇÃO DE SERVIÇOS

AAA CONSULTORIA EMPRESARIAL LTDA., pessoa jurídica de direito privado, inscrita no CNPJ: 00.000.000/0001-00, com endereço na Avenida Aaa Aaaaaa n. 000, Bloco 0, sala 000, Centro, Florianópolis, CEP 00.000- neste ato representada na forma de seu contrato social, de seu representante legal, doravante denominado simplesmente **CONTRATANTE**.

BBB TECNOLOGIA LTDA, doravante denominada **BBB** ou **CONTRATADA**, pessoa jurídica de direito privado, inscrita no CNPJ sob o nº 00.000.000/0001-00, com sede estabelecida em Florianópolis - SC, na Rua Bbbbbb Bbbbbbbb, nº 00, sala 000, bairro Centro, CEP 00000-000, neste ato representada na forma de seu contrato social, por seu representante legal.

Têm entre si, de maneira justa e acertada, o presente Contrato de Prestação de Serviços, ficando, desde já, aceito pelas cláusulas a seguir descritas e pelos documentos anexos que, devidamente assinados pelas partes, passam a integrar o presente contrato.

CLÁUSULA 1ª DO OBJETO e PRAZO

1.1. O objeto do presente contrato é o fornecimento de serviços de tecnologia da informação, pela **CONTRATADA** à **CONTRATANTE**, com modificações de um aplicativo utilizando os módulos abaixo como escopo:

Aplicativo

- Retirar os 3 card da tela inicial.

- Adaptar o aplicativo para carregar conforme o cadastro da empresa do usuário logado.

- Definir quais informações dinâmicas serão carregadas no aplicativo, por exemplo, labels, cores e outros. Apontar no aplicativo onde deve ser alterado.

- Backend do app utiliza Go como linguagem de programação, como será necessário criar um endpoint para retornar as informações dinâmicas da base de dados, pode ocorrer algum empecilho devido a falta de conhecimento na linguagem.

Portal empresa

- Criar CRUD para salvar informações administrativas para apresentar no aplicativo.

- Definir quais informações serão salvas nesta tela.

1.2. Os serviços deverão ser executados de forma remota conforme previamente acordado pelas partes na proposta.

1.3. A **CONTRATADA** prestará os serviços à **CONTRATANTE** através de seus sócios ou colaboradores, cabendo-lhe exclusivamente a escolha e designação das pessoas a serem alocadas à tarefa a ser executada, desde que conhecedores e especialistas do assunto em questão.

1.4. O presente contrato se dá em caráter de não exclusividade, restando ressalvadas as condutas de confidencialidade e demais termos específicos ora previstos.

1.5. O início do projeto está previsto para 15 dias após a assinatura do contrato, e a entrega prevista para 30 dias após o início, quando serão entregues as alterações do aplicativo.

1.6 Está previsto somente o treinamento do responsável interno do produto, bem como se a **CONTRATANTE** resolver substituí-lo deverá arcar com os custos de treinamento de outro responsável.

CLÁUSULA 2ª DAS CONDIÇÕES DE PRESTAÇÃO DOS SERVIÇOS

2.1. Os serviços serão prestados pela **CONTRATADA**, por telefone, ou internet, de acordo com a necessidade apresentada em cada caso.

2.2. Caso a **CONTRATADA** constate a impossibilidade de prestação dos serviços à distância, será agendada uma visita à sede ou filiais da **CONTRATANTE**, mediante reembolso das despesas de viagem, hospedagem e alimentação por parte da **CONTRATANTE** mediante aprovação prévia por parte da **CONTRATANTE**.

2.3. São obrigações da **CONTRATADA**:

I- Prestar os serviços ora contratados com estrita observância das condições fixadas de comum acordo e respeitando todas as prerrogativas de qualidade especificadas;

II- Tomar todas as cautelas necessárias para a perfeita execução dos serviços, evitando danos ou paradas/interrupções aos serviços da empresa **CONTRATANTE**;

III - Responder pela guarda e conservação de quaisquer equipamentos, materiais, informações de propriedade ou titularidade da **CONTRATANTE** ou de seus clientes que eventualmente lhe tenham sido acessíveis em decorrência ou para a prestação de serviços ora acordada, comprometendo-se a responder, na forma da lei, pelos danos ocasionados à **CONTRATANTE**;

IV - Desenvolver suas atividades, nos horários normais de funcionamento da **CONTRATANTE** ou nos casos necessários, fora do horário de expediente, objetivando o cumprimento das atividades pactuadas neste contrato;

V- Responsabilizar-se pelos atos cometidos por seus colaboradores;

VI - Manter a guarda dos documentos referente à execução do objeto do presente contrato, bem como a documentação que lhe for entregue pela **CONTRATANTE** para o desenvolvimento de suas funções, com a consequente obrigação de custódia e de utilizar tal documentação exclusivamente para o desenvolvimento de suas atividades;

VII - Enviar juntamente com a nota fiscal/fatura, relatório das horas prestadas no mês de referência, contendo descrição detalhada das atividades desempenhadas e cronograma do projeto em execução;

2.4. A **CONTRATADA** organizará suas atividades profissionais e o tempo dedicado à mesma conforme negociado com cliente – CLAUSULA 3ª, item a.

2.5. A CONTRATANTE, por seu turno e para a boa prestação dos serviços, obriga-se a:

- I** -Colocar à disposição da **CONTRATADA** todos os dados e instruções de que esta necessitar para consecução do aplicativo objeto deste contrato;
- II** -Permitir à **CONTRATADA** o livre acesso a toda infraestrutura indispensável para a consecução do objeto deste contrato, tanto remotamente aos computadores da **CONTRATANTE** como fisicamente na sede ou filiais desta;
- III** -Efetuar o pagamento dos valores ajustados neste contrato dentro dos prazos estipulados, conforme proposta em anexo.
- IV** -Indicar um colaborador responsável pelos contatos de ordem técnica com a **CONTRATADA**.

CLÁUSULA 3ª DO PREÇO E DO PAGAMENTO

3.1. Pela prestação efetiva dos serviços ora contratados, a **CONTRATANTE** pagará à **CONTRATADA** a importância prevista nos instrumentos: **ANEXO I - PROPOSTA COMERCIAL**, na forma abaixo:

a) Desenvolvimento:

Profissional	Horas	VI Unit	VI Total
Designer	35	R\$ 120,00	4.200,00
Análise	61	R\$ 105,00	6.405,00
Desenvolvimento	72	R\$ 100,00	7.200,00
Testes	22	R\$ 90,00	1.980,00
Gerenciamento	28	R\$ 130,00	3.640,00
TOTAL	218		23.425,00

3.2. FORMA DE PAGAMENTO.

O pagamento do serviço desenvolvimento será realizado 50% de entrada na assinatura do contrato e 50% trinta dias após o início dos trabalhos.

3.3 A responsabilidade com criação e manutenção das contas Apple e Google é da **CONTRATANTE**, bem como o custeio integral deste serviço é por conta exclusiva da **CONTRATANTE**.

3.4. Os valores da hospedagem serão corrigidos monetariamente pelo INPC, anualmente, sempre no aniversário do contrato.

3.5. Salvo quando do atraso na apresentação das respectivas notas fiscais/faturas, as parcelas não liquidadas nos respectivos vencimentos, ficarão sujeitas à multa de 2% (dois por cento), além de atualização monetária, pelo índice de correção pelo INPC estabelecido neste contrato, "*pro rata dia*",

acrescido de juros de 1 % (um por cento) ao mês, calculado sobre o débito já corrigido.

CLÁUSULA 4ª VIGÊNCIA E EXTINÇÃO

4.1. O presente contrato entrará em vigor a partir da data de sua assinatura, tendo validade até a conclusão do aplicativo e mais o período de garantia – 90 dias após a entrega.

4.2 Após este período, a sustentação e manutenção serão responsabilidade da CONTRATANTE. Para a CONTRATADA atender por demanda, o valor da hora avulsa é de R\$ 180,00 cada, reajustável anualmente pelo INPC no aniversário do contrato.

4.3 A CONTRATADA dispõe de um pacote de 20 horas mensais ao valor de R\$ 120,00 a hora, ficando a critério da CONTRATANTE a contratação e pagamento a parte, sempre com vencimento no dia 10 posterior ao trabalhado e reajuste anual.

4.4 Todas as informações confidenciais continuarão sendo de exclusiva propriedade da sociedade, não podendo nenhuma cláusula deste acordo ser interpretada como cessão de qualquer direito pertinente às informações confidenciais.

CLÁUSULA 5ª DA NÃO CONCORRÊNCIA

5.1. A CONTRATANTE e a CONTRATADA se comprometem, durante todo o prazo de vigência desta parceria, a não:

I. Participar, direta ou indiretamente em qualquer sociedade que atue ou esteja envolvida no mesmo mercado, seja como empregado, executivo, sócio ou acionista;

II. Contratar ou tentar contratar, bem como induzir, solicitar ou encorajar a contratação de qualquer empregado, prestador de serviço, executivo ou colaborador da sociedade;

III. Motivar, induzir, solicitar ou encorajar qualquer cliente, executivo, empregado, colaborador ou fornecedor da sociedade a cessar ou modificar sua relação comercial ou, ainda, interferir em seus negócios.

5.2. Caso o contrato seja rescindido antes do término da vigência desta parceria, a CONTRATADA compromete-se a respeitar esta Cláusula pelo prazo de 1 (um) ano a contar da data efetiva da rescisão.

CLÁUSULA 6ª DA CONFIDENCIALIDADE

6.1. A CONTRATADA se obriga por meio deste instrumento a manter sigilo sobre as informações confidenciais da parceria, tomando todas as medidas necessárias para que terceiros não tenham acesso às referidas informações.

6.2. Não serão consideradas informações confidenciais, para o propósito do presente instrumento, as informações que, comprovadamente:

I. Já eram de conhecimento público antes de sua divulgação pela sociedade;

II. Tenham chegado ao conhecimento público sem dolo ou culpa de qualquer uma das partes;

III. Devam ser divulgadas, por ordem judicial ou de autoridades competentes, sendo que o acionista alcançado por tal ordem deverá notificar previamente a sociedade acerca de sua existência;

Parágrafo Único. Todas as informações confidenciais continuarão sendo de exclusiva propriedade da sociedade, não podendo nenhuma cláusula deste acordo ser interpretada como cessão de qualquer direito pertinente às informações confidenciais.

CLÁUSULA 7ª DA PROPRIEDADE INTELECTUAL

7.1. Toda a propriedade intelectual desenvolvida pelas partes deste acordo, bem como por funcionários e prestadores de serviço envolvidos no processo, será de exclusiva titularidade da **CONTRATANTE**, possuindo todos os direitos, títulos, propriedades e licenças necessárias para a sua utilização na condução dos negócios, livre de quaisquer ônus, conforme a lei.

7.2. Todos os funcionários e prestadores de serviço deverão celebrar os instrumentos necessários para formalizar a cessão, em caráter irrevogável e irretratável, de toda a propriedade intelectual porventura desenvolvida por eles no desempenho de suas funções, isoladamente ou em colaboração com empregados, colaboradores, contratados ou demais administradores da sociedade, em favor da sociedade;

7.3. As partes se comprometem a tomar todas as medidas necessárias para assegurar os direitos da **CONTRATANTE** sobre a propriedade intelectual.

7.4. É vedado à CONTRATADA qualquer espécie de procedimento que implique engenharia reversa, descompilação, desmontagem, tradução, adaptação e/ou modificação de software, ou qualquer outra conduta que possibilite o acesso ao código fonte do software, exceto e somente na medida em que essa atividade seja expressamente permitida pela legislação aplicável, não obstante essa limitação.

CLÁUSULA 8ª DO REGIME DE PROTEÇÃO DE DADOS

8.1. A CONTRATADA se compromete a respeitar integralmente o regime legal de proteção de dados previsto na Lei 13.709/2018, garantindo a privacidade, sigilo, controle e segurança de todos os dados pessoais transmitidos pelos Titulares dos Dados Pessoais, a partir da contratação dos serviços da CONTRATADA, a fim de cumprir os termos estabelecidos no objeto descrito neste Contrato, protegendo-os de acessos não autorizados e de situações acidentais ou ilícitas de destruição, perda, alteração, comunicação ou qualquer forma de tratamento inadequado ou ilícito por parte exclusiva dos técnicos e profissionais da CONTRATADA.

8.2. A atuação da CONTRATADA como “Controladora Conjunta”, nos termos Lei nº 13.709/18 (Lei Geral de Proteção de Dados – LGPD), de eventuais dados pessoais e/ou dados sensíveis fornecidos pelos Titulares dos Dados Pessoais, observará todas as disposições determinadas na LGPD ou pela Autoridade Nacional de Proteção de Dados (ANPD).

8.3. A Parte que obtiver conhecimento de um incidente de segurança relacionado aos serviços objeto deste contrato deverá comunicar a outra Parte do incidente. Esta comunicação deverá conter:

- a) A descrição da natureza dos dados pessoais afetados;
- b) As informações sobre os titulares envolvidos;
- c) As medidas técnicas e de segurança utilizadas para a proteção dos dados;
- d) Os riscos relacionados ao incidente; e
- e) As medidas que foram ou que serão adotadas para reverter ou mitigar os efeitos do incidente.

8.4. A responsabilidade de comunicação aos Titulares dos Dados Pessoais, dentro do prazo legal, acerca de todo e qualquer incidente de segurança relacionado aos serviços deste Contrato será da Parte legalmente detentora dos dados, utilizando, para tanto, as informações fornecidas pela outra Parte.

8.5. A segurança no tratamento dos dados pessoais oriundos deste Contrato observará as técnicas atuais, os custos de aplicação, o contexto e a finalidade do tratamento, aplicando as medidas técnicas e organizacionais adequadas assegurando um nível de segurança adequado ao risco.

CLÁUSULA 9ª DOS IMPOSTOS

9.1. Todos os impostos, taxas e contribuições vigentes na data da formalização do presente instrumento e incidentes sobre a prestação dos serviços objeto do presente instrumento são de inteira responsabilidade da CONTRATADA.

Parágrafo Único. Fica desde já esclarecido entre as partes contratantes que, a responsabilidade por todo e qualquer novo tributo que venha a incidir sobre os serviços aqui previstos e/ou por todo e qualquer aumento dos tributos ora vigentes, será estabelecida pelas partes, oportunamente, de comum acordo.

CLÁUSULA 10ª DA INDEPENDÊNCIA SOCIETÁRIA E AUSÊNCIA DE GRUPO EMPRESARIAL

10.1. As partes, desde já, reconhecem entre si que a relação ora estabelecida pelo presente instrumento é meramente empresarial, sem quaisquer vínculos societários entre si, não constituindo, portanto, grupo empresarial de fato e/ou qualquer outra correlação societária.

Parágrafo Único. Em decorrência do disposto no caput desta Cláusula, as Partes se comprometem a manter independência societária, administrativa e de gestão ao longo da vigência do presente instrumento, arcando, cada uma das Partes, com suas obrigações e responsabilidades, sejam elas de qualquer origem.

CLÁUSULA 11ª DAS RESPONSABILIDADES

11.1. A **CONTRATADA** declara que goza de plena capacidade comercial e técnica, bem como que não está inabilitada nem incurso em causa de incompatibilidade legal para exercer as atividades definidas no presente contrato.

11.2. A **CONTRATADA** será a única e exclusiva responsável pelo pessoal que contratar para o desenvolvimento ou realização dos serviços objeto deste contrato, devendo esta responder por todos os recolhimentos fiscais, previdenciários, trabalhistas, securitários e quaisquer reclamações ou indenizações devidas ou ocasionadas pelos mesmos, seja qual for sua natureza.

11.3. Responsabiliza-se a **CONTRATANTE** pela origem e veracidade de todas as informações e documentação encaminhadas para a **CONTRATADA**.

11.4. Salvo em se tratando de vício oculto, a **CONTRATANTE** é responsável por notificar o mais brevemente possível a **CONTRATADA**, quando da ocorrência de alguma anormalidade, restando ainda ciente que a demora injustificada em tal notificação poderá ocasionar danos de sua inteira responsabilidade.

CLÁUSULA 12ª DISPOSIÇÕES GERAIS

12.1. A omissão ou tolerância, por qualquer das partes, em exigir o estrito cumprimento dos termos e condições deste contrato, não constituirá novação ou renúncia dos direitos aqui estabelecidos, que poderão ser exercidos plena e integralmente a qualquer tempo.

12.2. A presente contratação cancela e torna sem nenhum efeito todos os ajustes verbais ou escritos anteriormente firmados entre as partes a respeito do objeto deste contrato.

12.3. As partes reconhecem que não têm autoridade ou poder para, direta ou indiretamente, obrigar, negociar, contratar, assumir débitos, obrigações ou criar

quaisquer responsabilidades em nome da outra parte, sob qualquer forma ou com qualquer propósito.

12.4. A contratação do fornecimento de serviços é realizada em caráter não exclusivo, podendo a **CONTRATANTE** tomar as atividades aqui contempladas de quaisquer outros fornecedores, bem como a **CONTRATADA** prestar serviços a outros clientes, sem que se configure infração a este instrumento, desde que observadas as condições de confidencialidade ora estabelecidas.

12.5. Este contrato não poderá ser cedido, no todo ou em parte, ressalvada a concordância expressa e por escrito de ambas as partes.

12.6. O disposto neste contrato não poderá ser alterado ou emendado pelas partes, a não ser por meio de aditivos dos quais conste a concordância expressa e por escrito das partes.

12.7. As comunicações entre as partes poderão se dar através de correio eletrônico entre contas corporativas das partes, tomando-se este meio eletrônico como documentação válida entre as partes, quando comprovadamente enviadas e recebidas pelas mesmas.

CLÁUSULA 14ª DO FORO

13.1. Elege-se o Foro da Comarca de Blumenau – Santa Catarina para dirimir eventuais questões atinentes a este contrato em que a intervenção judicial se faça necessária.

E por estarem assim justas e acertadas, as partes firmam o presente instrumento, em 02(duas) vias de igual teor e forma, tudo na presença das testemunhas abaixo.

Florianópolis – Santa Catarina, 00 de abril de 0000.

COTRATADA

CONTRATANTE

Testemunhas:

1 _____
Nome:
CPF:

2 _____
Nome:
CPF: