Evaluating Complexity and Digitizability of Regulations and Contracts for a Blockchain Application Design.

Pradeepkumar D S, Kapil Singi, Vikrant Kaulgud, Sanjay Podder
Accenture Labs, Bangalore, India
{p.duraisamy,kapil.singi,vikrant.kaulgud,sanjay.podder}@accenture.com

ABSTRACT

Blockchain technology becomes the key solution to provide trust and security without any need for a central supervisory authority to validate the transactions. By now, it plays a key role in the digital transformation of several processes and industries with varying application use cases. To promote the wide adoption of blockchain technology we need mechanisms to identify the digitizability level of the given regulations to smart contracts and mechanisms to specify which blockchain technology is best suitable for the given regulations. In this work, we propose a modeling approach that supports the automated analysis of human-readable regulation representations by suggesting how much percentage of regulation is digitizable and the suitable blockchain environment to design the application. We identify smart contract components that correspond to real-world entities and its pertaining clauses and its digitizability property. With selected examples, we explore this capability and discuss our future research directions on smart contract generation according to the recommended environment.

CCS CONCEPTS

• **Software and its engineering** → *Software design engineering*;

KEYWORDS

Blockchain, Regulations, Digitizability Complexity, Smart Contracts

ACM Reference Format:
Pradeepkumar D S, Kapil Singi, Vikrant Kaulgud, Sanjay Podder. 2018. Evaluating Complexity and Digitizability of Regulations and Contracts for a Blockchain Application Design.. In *WETSEB'18: WETSEB'18:IEEE/ACM 1st International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB 2018), May 27, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, Article 4, 5 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

Blockchain technology is becoming the defacto standard for industrial applications, as technologies and business approaches get distributed in virtually all digitalized areas.

With conventional centralized data storage model, the central owner keeps a copy of the ledger. Consequently, this entity controls the participants and permissions across the network. With the advent of distributed ledger technologies (DLT), there will be multiple entities holding copy of the underlying ledger and all participants are naturally permitted to contribute. In DLT, all participants agree upon a common truth, e.g. the correctness of a ledger, as changes made by one node have to be propagated to all other participants in the network. The result of arriving at a common truth is called consensus among participants.

The consensus can be categorized into two based on the mode of operation: permissionless and permissioned. In permissionless, every participants in a network can see all the transactions and can monitor the progress of the transaction. This blockchain model is called the public blockchain [4]. On the other hand, Fabric [6] and Corda [7] restricts participants with permissions, for example, restricting a transaction visible to few participants rather all.

Blockchain technology is being tested and implemented across a broad range of applications, industries with various use cases across multiple domains. Blockchain technology tackles the key problems inherent in a business transactions, like Trust, Transparency and Accountability. Trust - Blockchain uses cryptography to validate the trust among the participants. Transparency - since the ledger is distributed, all peers involved in the transaction network can view it (permissioned model enforce additional security rights); Accountability - Since all parties in the transaction can view the distributed ledger, everyone can validate the transaction and can agree on the same.

One of the most critical part of blockchain development is writing smart contracts to define rules and penalties around an agreement and enforcing those obligations. Smart contracts are autonomous stateful scripts that are stored in the blockchain and provide on-demand execution and validation of requests. In Ethereum smart contracts can be written in Solidity[1]/Serpent[2]/LLL[3] languages. In Hyperledger Fabric the smart contract (known as Chaincode) can be written in golang, Javascript and Java.

From our practical experience, the process of codifying a regulation document (a.k.a. Service Level Agreement (SLA)) to smart contract requires nearly 3 to 4 months of team effort, and only after such efforts the team will conclude whether the given regulations can be automated or require refinements for automation. Also identifying suitable blockchain infrastructure based on regulations is another challenge driven by regulation policy.

We believe that creating mechanism to automate the classification of regulations into (1) automatable; (2) semi-automatable; (3) manual, and further inferring the suitable blockchain infrastructure to be used from the given regulations document, can minimize the human effort and helps leverage the development process life cycle.

Section 2 describe the related works, the Section 3 we discuss the digitizability of regulations followed by our approach and real-time

WETSEB, 2018,

ACM ISBN 123-4567-24-567/08/06…$15.00
https://doi.org/10.475/123_4

examples in Section 4. Finally, we end our paper with conclusion and future work (in Section 5).

2 RELATED WORK

The research work by Sheppered [12] states the software project failure due to illusory metric evaluation and the requirement for a more holistic and inter-disciplinary system. Kaner and Bond [11] proposes a multidimensional analyses of attributes through direct and indirect measurement through breakdown of an attribute into sub-attributes and grouping them for analysis.

The work by Misirli et al. [13] explains the strength of using correlation for code and network metrics by suggesting the metric system to consider the parameters related to field than the parameters related to code to get higher correlation with defect category. This helps us to use classification based on blockchain environment at first-class citizen than smart contract code generation mechanism.

Software can be measured at various level [14] like classes and interfaces, based on couplings, and abstractness. The work in [15] suggests how paying too much attention to one metric or another can lead to incorrect views of software systems. The work by Nagappan and Ball [16] analyse how the production systems can be evaluated post release and the impact of code dependencies.

The work by Massimo and Livio [8] analyses various blockchain environments and the impact of smart contracts on these platforms. The paper also studies usage of smart contracts to different domains and their usage patterns. Frantz and Nowostawski [9] introduce a mechanism of automatic smart contract generation from regulations. Our work makes an addition value to the processing of regulations by introducing the clause digitizability classification chart. The usage of cyclometric complexity in software testing was well explained by McCabe [10]. Cyclometric complexity helps to identify the code bugs and helps in predicting the complexity of the code. Similar to that, the proposed digitizability complexity helps to understand the complexity of the statement and hence it refrains complex codes before the design/development state starts.

3 DIGITIZABILITY OF REGULATIONS

Regulations document or SLA is signed between parties that define the level of business agreement made. SLA consists of contractual clauses (individual statements) that specifies the entities (participants), their obligations and constraints.

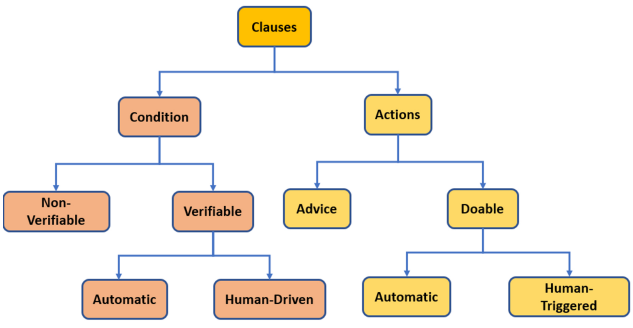


Figure 1: Clause Classification Chart

Based on our analysis on various regulations documents like GDPR regulations [19], rental agreements [18] etc., we derived a mechanism to classify the regulation's contractual clauses into two broad category, (1) Conditional Clauses and (2) Actionable clauses. The clause classification chart is shown in Figure 1. In the following sub-sections we will discuss in more detail about clause classification.

3.1 Conditional Clauses

Conditional clauses specify obligations that the participants has to agree upon, and the actionable clauses specify the actions to be done provided the specified conditions are satisfied.

The conditional clauses are classified as (a) Non-verifiable and (b) Verifiable conditions.

Non-verifiable Clauses present a heuristics or statement that require information from multiple source for validation (due to a lack of information about the objective on which validation can happen) and hence cannot be validated by a system automatically.

Example: *"The Borrower declares that the information and data furnished by it to the Lender are true and correct."*

Verifiable clauses represent a condition which can be verified by the system directly or indirectly. We classify the verifiability property into (a) Automatically verifiable, (b) Human-driven respectively.

3.1.1 Automatic Verification. Automatically verifiable statements represent a closure condition that a system can validate automatically. For example, the system can automatically validate the below statement as per the rules defined in Schedule 1.

Example: *"The Borrower shall pay the interest along with the principle and penal interest of penalties on 10th of every month."*

3.1.2 Human-driven Verification. Regulations can also have clauses that specify human-driven verification (i.e. human in the process), like in triggering events or workflow approval etc. For example, the following statement requires Lender (a human) to verify the Loan before processing to the Borrower.

Example: *"The Loan shall be disbursed by the Lender to the Borrower within a period of 7 (seven) days from the date of execution of this Agreement"*

3.2 Actionable Clauses

Moving on to action, the statement or part of a statement, that specify the obligations or rule to be triggered upon a successful condition. Actions are classified as (1) Advice, and (2) Doable action. **Advices** represent plain declaration that do not impact the system without further input from a human or other external source. For example, in the following statement, "he should be penalized" doesnot specify any mechanism or standard that a system can apply.

Example: *"The Borrower shall pay the interest as per the Schedule 1, in case, if the borrower missed to pay the interest then the Lender can send a notification."*

Doable Actions represents the direct and indirect automatable actions. We justify the doability of an action under two broad category (a) Automatic, and (b) Human-triggered.

3.2.1 Automatic Doable Actions. Doable actions can be triggered directly by the system without any human assistance. For example, in the following statement the system can fetch all the required information automatically like: loan amount, calculating 24% on the loan.

Example: *"Any default by the Borrower in payment for dues towards interest or principle would entail an additional interest charge of 24% on the entire Loan."*

3.2.2 Human-triggered Doable Actions. Regulations will have clauses that specify human to trigger an event (i.e. human in the process), similar to workflow triggers etc.

Example: *"The Loan shall be disbursed by the Lender to the Borrower within a period of 7 (seven) days from the date of execution of this Agreement"*

In the above statement disbursal of loan within 7 day period has to be calculated based on the agreement initiation time and a human agreement for initiation process.

3.3 Digitizability Score

Based on the above sentence classifications method, we formulate the digitizability complexity analysis at par with cyclometric complexity, which helps to identify the following from a regulation document:

- Does the statement codifiable?
- Does the statement represent closure with self-dependent components/entities?
- Does the document is reliable enough for blockchain?

The digitizability score of a document helps to understand the document reliablity for the blockchain environment.

Categories	Condition			Actions		
		Verifiable			Doable	
	Non-Verifiable	Automatic	Human-Driven	Advice	Automatic	Human-Triggered
Non-Digitizable	✓		✓	✓		✓
Semi-Digitizable		✓	✓	✓	✓	✓
Digitizable		✓			✓	

Figure 2: Digitizability Chart

The document classifier contains three broad categories, (1) non-digitizable statements; (2) semi-digitizable statements; and (3) digitizable statements, as shown in the Figure 2.

Digitizability complexity is a regulation document complexity measurement that is being correlated to a number of digitizable and semi-digitizable statement within it.

Regulation document composes of multiple regulation clauses. Regulation clauses can be written using simple, compound, or complex sentences. Generally, from our experience we identified that humans tend to write complex/compound statements. Writing a regulation statement in simple sentence is more important, since it will improve the maintainability and understandability. We assign complexity scores and probabilistic score (p) to the statements based on the following three broad categories. Firstly, if the document contains all digitizable statements (i.e. auto verifiable and auto doable), the complexity would be 1, since there would be only a single path through the code without human-triggered/human-verified events. The probabilistic value for this kind of statements are 1 (i.e. 100% automation possible). Secondly, if the statement had one part automatable and the other part human triggered/verified, there would be two paths in the generated contract code: one part of the statement code executes automatically in the blockchain and another part of the statement code requires human verification/trigger to evaluate the execution, so the complexity would be 2. The probabilistic value for this kind of statements are 0.5 (i.e. (50-50)%). Finally, if the statement contains non-verifiable condition or if the statement contains fully human verifiable condition or human triggered actions, would produce a complexity of 3. The probabilistic value for this kind of statements are 0 (i.e. 0% automatable part present). Formally, the probabilistic score of a statement ($p(s)$) can be defined as follows:

$$p(s) = \begin{cases} 1 & \text{if } s \text{ is fully automatable} \\ 2 & \text{if } s \text{ is (50-50)\% automatable} \\ 3 & \text{if } s \text{ is human verifiable/triggered} \end{cases}$$

The digitizability complexity is calculated by developing a directed graph of the statement that measures the number of linearly independent paths through a regulation statement. Similar to the cyclometric complexity (used to measure code complexity), we are representing the digitizability complexity of the document as a directed graph with nodes representing the statement and edges between two nodes representing the connection between statements. The digitizability complexity is used to derive a measurable value based on the number of edges and nodes within the graph as well as the total count of connected components or exit nodes. The digitizability complexity calculation for a simple sentence is represented as shown below:

$$Complexity(M_s) = E - N + 2P \quad (1)$$

where, M_s represents the digitizability complexity of a simple statement, E represents the number of edges of the graph, N represents the number of nodes of the graph, and P represents the number of connected components. In this paper, we consider each statement as a method or subroutine, and hence P is always equal to 1.

A complex or compound clause can be broken down into multiple simple sentences. The digitizability score for complex/compound clauses and the document can be calculated as follows:

$$Clause\ Complexity(M_c) = \frac{1}{n}\sum_{1}^{n} p(M_s) \tag{2}$$

$$Document\ Complexity(M_d)\ in\ \% = \sum_{1}^{n} M_c \times 100 \tag{3}$$

where, n represents number of simple statements, $p(M_s)$ represents the probabilistic score of the statement's digitizability complexity score, M_c represents the digitizability complexity of a clause, and M_d represents the digitizability complexity of a regulation document.

4 APPROACH

In this section we explain the system architecture, and the complexity calculation algorithm with example.

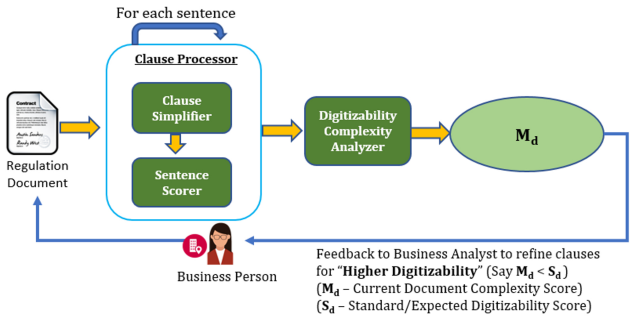


Figure 3: Digitizability Tool Architecture

The above figure 3 illustrates the system flow starting from regulation document and generating the digitization score. The 'Clause Simplifier' module reduces the clauses into simple sentences using Stanford's CoreNLP [5]. For example, lets see how a complex sentence is simplified into multiple simple sentences as follows:

COMPLEX SENTENCE: "*The Borrower shall pay the interest as per the Schedule 1, in case, if the borrower missed to pay the interest then the Lender can send a notification. And the transactions related to the loan amount should be disclosed only to the Borrower and the Auditor.*"

The presence of connection points like "after|although|if|..", represents a Complex Sentence. Breaking at connection points gives us three simple sentence chunks (as underlined) as follows:

SIMPLIFIED FORM: *The Borrower shall pay the interest as per the Schedule 1. If the borrower missed to pay the interest then the Lender can send a notification. The transactions related to the loan amount should be disclosed only to the Borrower and the Auditor.*

Below we show our clause processor algorithm (Figure 4) on how to simplify the given complex/compound clauses into simple sentences and how the classification of automatable or human-driven sentences were handled. The algorithm follows 'The Penn TreeBank Tagset' [17] for sentence classification.

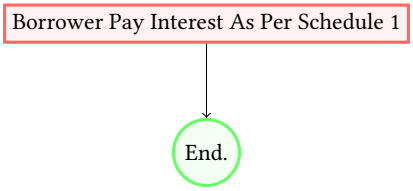
Algorithm ***(Clause Processor)***:

(1) Take input clause from the contract document.
(2) Convert the sentence into simple form by identifying the presence of connection words as follows:
  (a) If the given clause contains connections like *"after | although | as | because | before | even though | if | since | though | unless | until | when | whenever | whereas | wherever | while"*, then it is Complex Sentence; GoTo 3.
  (b) If the given clause contains connections like *"For | And | Nor | But | Or | Yet | So"*, then it is Compound Sentence; GoTo 4.
  (c) If the sentence is Simple Sentence; GoTo 5
(3) Chunk the sentence at the connection point and do the following:
  (a) If chunk sentence contains conditions, associate it with the previous chunk
  (b) Chunk sentence should have at least one Noun and Verb associated with it.
  (c) Upon successful chunk of Complex sentence into bag of sentences, GoTo 4.
(4) Chunk the sentence at the connection point and do the following:
  (a) Chunk sentence should have at least one Noun and Verb associated with it.
  (b) Upon successful chunk of Compound sentence into bag of sentences, GoTo 5.
(5) Extract each sentence from the bag of sentences, **(Clause Classification)**:
  (a) Parse the sentence using Stanford NLP Parser, extract all the Nouns and Verbs
  (b) If the sentence contains verbs (VBZ or VBN) - Tag them as **Non-verifiable sentences**
  (c) Find the presence of Modal Verbs like *"Must | Shall | Will | Should | Would | Can | Could | May | Might"*.
    (i) If the sentence contains Modal Verbs - Tag them as **Human-driven Sentences**
  (d) If both (b) and (c) conditions are present - Tag them as **Human-Verifiable & Human-driven Sentences**
  (e) Presence of Verb (VB) and Noun (NNP) will give a proper simple sentence - Tag the sentence as **Automatic**.
(6) From the classified sentence, extract the Noun, Verb, and Associations.
(7) Form a set that is represented as follows ClauseID, Noun, Verb, Associations.

Figure 4: Algorithm

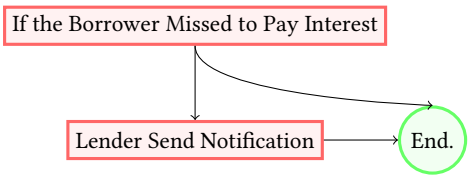
Lets calculate the digitizability score of the above sentence as follows.

Part 1: The Borrower shall pay the interest as per the Schedule 1.



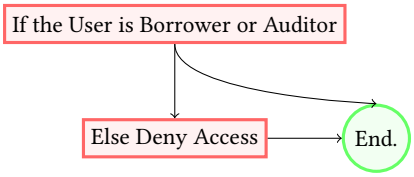
We have 2 nodes (N), 1 edge (E), (M_s) is 1 (i.e. M_s = 1 - 2 + 2).

Part 2: If the borrower missed to pay the interest then the Lender can send a notification.



(M_s) is 2 (i.e. M_s = 3 - 3 + 2).

Part 3: The transactions related to the loan amount should be disclosed only to the Borrower and the Auditor.



M_s is 2 (i.e. M_s = 3 - 3 + 2).

Finally, the 'Sentence Scorer' will give the digitizability score at statement level, and the 'Digitizability Complexity Analyzer' will measure the digitizability complexity for the document using the Formula (M_d).

M_c is 0.66 (i.e. M_c = (1 + 0.5 + 0.5)/3).

M_d is 66% (i.e. M_d = 0.66 × 100). (for one clause)

M_d suggest the percentage of digitizability. In case, if the system has a preconfigured threshold value say 70%, then the digitizability score of the document becomes less than the configured system value, and hence the system will suggest the Business person to rework on the clauses to increase the digitizability complexity score else if the score is above threshold value then it will proceed to development phase.

5 CONCLUSION AND FUTURE WORK

In this paper, we present a mechanism to measure the digitizability complexity of a regulation document and to identify the digitizability level of statements to smart contracts. We designed a novel digitizability complexity metrics that helps the software development team to measure the reliability of modules and measuring the testing difficulties of the given regulations in the SLA document by giving digitizability score and suggesting which blockchain environment is most suitable.

The digitizability score helps the consultants to understand the coding and testing complexity involved in the regulations, and thereby they can work at the initial stage of the regulations to reduce the complexity before the development process begins.

As a future work, we plan to extend this model to test the smart contracts to identify the metric equivalence between cyclometric and digitizability complexity method.

REFERENCES

[1] Solidity. https://github.com/ethereum/solidity. Accessed on 02/1/2018.
[2] Serpent. https://github.com/ethereum/wiki/wiki/Serpent. Accessed on 02/1/2018.
[3] LLL. http://lll-docs.readthedocs.io/en/latest/lll_reference.html.
[4] Ethereum. http://www.ethdocs.org/en/latest/. Accessed on 02/1/2018.
[5] https://stanfordnlp.github.io/CoreNLP/. Accessed on 02/1/2018.
[6] Hyperledger Fabric. https://github.com/hyperledger/fabric. Accessed on 02/1/2018.
[7] Mike Hearn, Corda: A distributed Ledger, "https://docs.corda.net/_static/corda-technical-whitepaper.pdf", November 2016.
[8] Massimo Bartoletti and Livio Pompianu, "An empirical analysis of smart contracts: platforms, applications, and design patterns", inâĂĆLecture Notes in Computer Science, March 2017.
[9] Christopher K. Frantz, Mariusz Nowostawski, "From Institutions to Code: Towards Automated Generation of Smart Contracts", 2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS*W), Augsburg, 2016, pp. 210-215.
[10] T. McCabe, "Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric," NIST Special Publication, (1996) September, pp. 500-235.
[11] Kaner, C., Bond, W. "Software Engineering Metrics: What Do They Measure and How Do we Know?", 10th International Software Metrics Symposium, 2004.
[12] Mair, C., Shepperd, M., "Human Judgement and Software Metrics: Vision for the Future" WETSoMâĂŹ11, 2011.
[13] Misirli, A., Caglayan, B., Miransky, A., Bener, A., Ruffolo, N, "Different Strokes for Different Folks: A Case Study on Software Metris for Different Defect Categories", WETSoMâĂŹ11, 2011.
[14] Robert Cecil Martin, "Agile Software Development: Principles, Patterns and Practices", Pearson Education, 2002.
[15] http://www.codeproject.com/Articles/28440/When-Why-and-How-Code-Analysis
[16] Nagappan, N., Ball, T., "Using Software Dependencies and Churn Metrics to Predict Field Failures: An Empirical Case Study", IEEE 1st International Symposium on Empirical Software Engineering Measurement, 2007.
[17] https://gist.github.com/nlothian/9240750. Accessed on 10/3/2018.
[18] https://www.tenancy.govt.nz/forms-and-resources/?keyword=&topic=&target=&action_updateFilter=Search&type=All
[19] https://ec.europa.eu/info/law/law-topic/data-protection_en