A Controlled Natural Language to Support Intent-based Blockchain Selection

Eder J. Scheid, Patrick Widmer, Bruno B. Rodrigues, Muriel F. Franco, Burkhard Stiller
Communication Systems Group CSG, Department of Informatics IfI, University of Zürich UZH
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland
[scheid,rodrigues,franco,stiller]@ifi.uzh.ch, patrick.widmer@uzh.ch

***Abstract*—In the last years, cryptocurrencies have become increasingly popular along with their underlying distributed ledger technology, referred to as a Blockchain (BC). Nowadays, a wide variety of BC implementations are available. However, the selection of a suitable implementation for a particular application or use case is complex because it requires technical understanding of the underlying BC implementation aspects. Therefore, this paper proposes a Controlled Natural Language (CNL) to extends existing BC selection solutions to abstract underlying implementation details. The approach allows the specification abstract high-level policies, referred to as intents, in an English-based language. The approach is inspired by previous approaches from the network management field. Moreover, a state machine-based refinement technique is proposed to refine these intents into low-level BC selection policies. The results of the performance evaluation of the prototype implementation show that the refinement process presents a minimal overhead. In addition, the perceived intuitiveness of the CNL by users was assessed in a survey. The results of the survey suggest that technical and non-technical individuals benefit from an intent-based approach equally.**

***Index Terms*—Intent; Blockchain; Selection; Controlled Natural Language.**

I. Introduction

The Blockchain (BC) concept gained popularity due to high speculation over the value of its underlying cryptocurrencies. Bitcoin, the most prominent BC implementation, was released in 2009 [13], and, since then, more than 4.900 cryptocurrencies, BC implementations and tokens have arisen [5]. However, even with the popularization of the BC concept and the media attention, most of the theory behind its technical aspects and interaction mechanisms are still not fully understood by many of its users.

Moreover, a single BC implementation might not fulfill all the requirements and Service Level Agreements (SLA) of a particular use-case. Taking the pharmaceutical cold-chain [2] as an example, where measurements of drugs temperature during transport are stored in the BC to ensure quality, the integrity of the data should be maintained while minimizing transaction costs. However, if an SLA from a client states that data must be stored in the BC in less than x minutes, transaction costs are disregarded over selecting a BC that complies with the SLA. Mapping these requirements to a particular BC is not a trivial task without the knowledge of technical details from available BC implementations. Therefore, there is a need to provide an interface where users define their BC selection requirements (*i.e.,* queries) in a less complex and abstract manner closer to Natural Languages (NL).

In addition, one novel concept that aims to abstract technical requirements from low-level configurations is Intent-based Networking (IBN) [8], [20]. In IBN, network operators define their intents, which are abstract, high-level policies used to operate the network [1], to guide the operation of the network. Intents define expectations of the operator, *i.e.,* "what" the network should perform and not "how" it should be achieved, without specifying technical details. Thus, the concept of intents can be applied in the BC selection context to allow users to specify what is their expectation from the selected BC without knowing specific implementation details.

The usability of NLs query languages and their representations have been studied in the literature, suggesting that "superior user support can be achieved by imposing some restrictions on the user's natural language input to guide the query" [10]. In this sense, Controlled Natural Languages (CNL), which restrict the user input to a defined grammar, can be employed to guide BC selection queries. Further, CNLs were widely applied in areas, such as network management [12], to reduce the complexity of tasks from network administrators when defining low-level rules. Thus, CNLs are able to represent the intent of the user during the BC selection process while reducing technical complexity in such an interaction.

Thus, this paper proposes a CNL in which non-technical users specify their abstract high-level requirements regarding BC selection in the form of an intent. The approach is supported by a state machine-based refinement method that translates these intents into low-level BC parameters (*e.g.,* block time, deployment type, and data size constraints), which can be utilized by policy-based BC selection frameworks, such as [14], to automatically select the BC that matches with the refined low-level parameters. In summary, the contributions of this work are *(i)* a BC selection CNL based on English, and *(ii)* a state machine-based intent refinement approach.

The remainder of this paper is structured as follows. Section II overviews related work concerning intent-based approaches. Then, Section III presents the CNL and the refinement technique along with their implementations. Further, Section IV evaluates the refinement technique and the intuitiveness of the CNL. Finally, Section V summarizes the paper and presents future work to be researched.

II. Related Work

Intent-based Management (IBM) is relatively an infant research area. However, this concept is being applied and researched in different contexts, such as network management [7], [8], [15], [18] and cloud management [3], [9], and discussed by the Internet Engineering Task Force (IETF) in several Internet-Drafts [4], [11], [17]. It should be noted that these Internet Drafts are valid for a short period of time and may replaced or become obsolete. Nevertheless, they are an indication of the growing interest and research in the intent topic.

A. Network Management

INSpIRE [15] proposes a refinement technique that translates intents into a set of configurations to create service chains in Network Function Virtualization (NFV)-enabled environments. It supports both homogeneous environments that consist of solely Virtual Network Functions (VNFs), and heterogeneous environments that consist of VNFs and physical middleboxes. The work determines, based on a defined Softgoal Interdependency Graphs (SIG) and clustering techniques, the VNFs that are required to fulfill an intent, chains these VNFs based on their dependencies, and presents low-level details to network devices for posterior traffic steering.

In [7], is proposed an intent-based approach to manage Virtual Networks (VNs) based on Software-defined Network Virtualization (NV). The goal of the approach is to automate the management and configuration of VNs based on intents, *i.e.,* high-level requirement specifications. The implementation is based on an open-source Software-Defined Networking (SDN) controller. Moreover, the approach is able to provide multiple VNs over the same physical infrastructure.

Further, the creation of secure SDN orchestration services based on intents is proposed in [18]. The approach relies on SDN orchestration and intents to automate the configuration and deployment of secure services. In an experimental evaluation, the authors found that the overhead of the approach is negligible, but the setup phase time for the supported technologies must be considered.

An extension of IBN has been proposed in [8]. The approach enhances the refinement process by employing machine-learning techniques and feedback from the network operators. In the approach, network operators interact with a chatbot, which adapts itself to unknown inputs. The work introduces an intermediate representation extracted from intents specified in NL, and present a prototype that translates intents from NL into the defined intermediate representation, and finally, into low-level network configuration rules.

In [6], the authors describe an intent-based approach to discover and deploy networks. The intents use a verb-object-subject sentence structure and are specified as tuples. The refinement approach utilizes so-called Maat agents to mediate between user intents and policies of network operators. There are still open challenges for the proposed approach, such as scalability of the deployment and security-related aspects.

TABLE I: Overview of Related Work

Work	**Intent Specification**	Refinement	**Area**
[15]	CNL	SIG and clustering	Network management
[8]	Natural language	Machine learning	Network management
[7]	Natural language	Multi-layer translation	Network management
[18]	JSON	N/A	Network management
[6]	Tuples	Ontology	Network Management
[3]	Natural language	Linear regression	Cloud management
[9]	Natural language	Label trees	Cloud management
This Work	CNL	Lookup tables	Blockchain

B. Cloud Management

In [3], the authors propose an intent-based approach to cloud service management. The objective is to automate or support the decision-making process for cloud resources of cloud operators. The approach accepts the requirements of cloud users as intents in a declarative manner and decides the composition and volume of the resources parameters.

In [9], the authors propose an intent-based approach to manage cloud infrastructure. The approach distinguishes between intents from the underlying infrastructure implementations. In contrast to existing systems that resolve conflicts during runtime, the approach detects and resolves conflicts already during the specification. The approach is called label management service and automatically creates label namespaces based on data from the cloud infrastructure.

C. Discussion

In the network management area, there are intent-based approaches that rely on NL for intent specification. However, the refinement techniques of these approaches are more complex than the refinement described in this paper. The refinement of [15] relies on softgoals and operationalizations, which are manually defined. The refinement process described in [8] relies on machine learning techniques using a chatbot as user interface. The platform described in [7] includes a refinement process and enforces the refined policies, resulting a more a more complex but complete process. In contrast, there are approaches in network management that do not rely on NL to specify intents. Specifically, intents in [18] are specified as a set of key-value pairs in the JSON format. In the cloud management area, current underlying refinement processes are also more complex than the refinement proposed herein. For instance, the refinement process described in [3] employs linear regression techniques to calculate parameters. Similarly, the refinement process used in [9] relies on complex label trees. Table I provides an overview of the similarities and differences of such approaches.

In summary, there are various intent-based approaches in network management and in cloud management, in which most allow intents to be specified in NL. However, their refinement processes differ and are tied to the specific use-cases. Therefore, these approaches cannot be directly applied to a novel context, such as BC selection. However, it is clear that the academia is focused on the research on the employment of intents in several areas and that such a concept is a promising research direction.

III. BC Selection Intent Language

The overview of the intent refinement process proposed herein is depicted in Figure 1. Intents are input by users in a CNL grammar, which is presented in Section III-A. The grammar allow users to compose their intents using pre-defined parameters (*cf.* Section III-B) to guide the BC selection. Once an intent is composed, it is refined (*i.e.,* translated and validated) to low-level BC selection policies based on the state-machine presented in Section III-C.

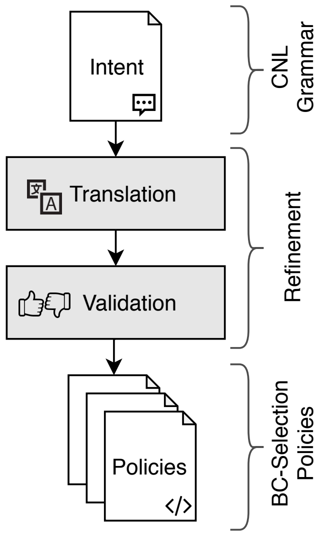

Fig. 1: Refinement Overview

A. Intent Grammar

Grammar 1 specifies the grammar of the intent language using the Extended Backus-Naur Form (EBNF) [16]. EBNF is an extension of the Backus Naur Form (BNF), which additionally supports the specification of options and repetitions. BNF and EBNF are notation techniques that allow the formal specification of context-free grammars. They are often used to specify the syntax of programming languages. Moreover, commas indicating sequences are omitted, for clarity. Further, trailing semicolons or dots indicating the end of the right-hand-side expression are not used. Similarly, whitespaces are omitted from the specification. Instead, every symbol in a sequence is implicitly separated by whitespace. Based in this CNL, users are able to author simple and complex intents for a single client or for multiple clients (*cf.* Table III).

B. Intent Parameters

An intent can be composed by different parameters, which should be supported by the Policy Decision Point (PDP) in the framework in which the policy is going to be employed.

Table II presents the classification of the available options based on their usage. *Conditions* are used to select an active policy from a set of policies. *Selection* strategies are used to select a BC from the active policy to store incoming transactions. *Filters* are used to restrict the pool of BCs.

```
⟨intent⟩       ::= for ⟨users⟩ [ in the ⟨timeframe⟩ ] select (the
                   ⟨profile⟩ [ ⟨filters⟩ ] blockchain [ ( from | except)
                   ⟨blockchains⟩ ] | ⟨blockchain⟩) [ with ⟨modifiers⟩ ]
                   ( until the ⟨interval⟩ costs reach [ ⟨currency⟩ ]
                   ⟨threshold⟩ | as default )
⟨users⟩        ::= ⟨user⟩ ( , | and ) ⟨user⟩
⟨user⟩         ::= [a-z0-9]+
⟨timeframe⟩    ::= day | night | morning | afternoon
⟨profile⟩      ::= cheapest | fastest
⟨filters⟩      ::= ⟨filter⟩ ( , | and ) ⟨filter⟩
⟨filter⟩       ::= private | public | fast | cheap | stable |
                   popular
⟨blockchains⟩  ::= ⟨blockchain⟩ ( , | and ) ⟨blockchain⟩
⟨blockchain⟩   ::= Bitcoin | EOS | Ethereum | Hyperledger | IOTA
                   | Multichain | Stellar
⟨modifiers⟩    ::= ⟨modifier⟩ { ( , | and ) ⟨modifier⟩ }
⟨modifier⟩     ::= encryption | redundancy | splitting
⟨interval⟩     ::= daily | weekly | monthly | yearly
⟨currency⟩     ::= CHF | EUR | USD
⟨threshold⟩    ::= integer | real
```

Grammar 1: BC Selection Intent

Filters can be further divided into static and dynamic options. Static filter options, such as the deployment type of a BC, remain constant over time. Dynamic filter options, such as transaction cost, change over time. *Modifiers* are used to alter the filtering or selection process. Most of the configuration options are directly mapped onto options available in the low-level policies. Options, *e.g.,* the timeframe, provide an abstraction of the options from the low-level policies. There are options, *e.g.,* `encryption` which are not available in the low-level policies. Finally, there are options, *e.g.,* `turing complete`, which are only available in the low-level policies.

Although intents are specified in NL, the negation of options is not supported. Negations could be implemented only for a subset of the available options, *e.g.,* `private`. The negation of `private`, *i.e.,* `not private` is equivalent to `public`. Similarly, all the options that could be negated can already be specified without negation. Therefore, it was decided to not support negation of options to avoid unnecessary complexity. Each of these options can have different parameters.

TABLE II: Classification of Configuration Parameters

Usage	**Parameters**
Condition	Users, Timeframe, Cost Interval, Cost Currency, Cost Threshold
Selection	Profile, BC
Filter	Deployment Type, Transaction Rate, Transaction Costs, Maturity, Whitelist, Blacklist
Modifier	Redundancy, Encryption, Splitting

Conditions can specified by users, *i.e.,* actor interacting with the system that may define one or more intents, in terms of:

- **Timeframes**: time-interval in which a refined intent (*i.e.,* policy) is active, *e.g.,* day or night.
- **Cost interval**: interval (*e.g.,* daily, weekly, monthly, yearly) at which rate accumulated transaction costs are reset.

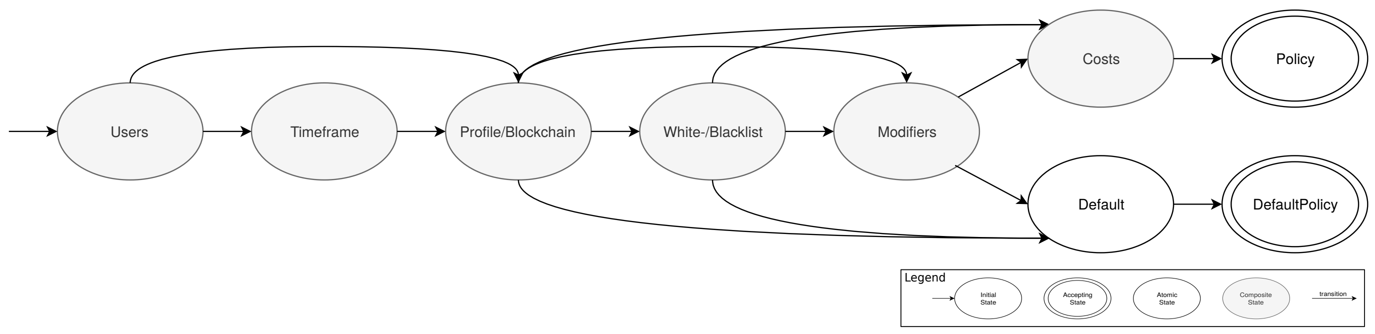

Fig. 2: Intent Parsing State Chart Overview

- **Cost currency**: currency of the specified cost threshold (*e.g.,* `CHF`, `EUR`, or `USD`).
- **Cost threshold**: cost threshold for the specified interval.

Selections can be composed of **profile** or **BC**. Generally, the specification of a profile is required and the specification of a BC is optional. Moreover, they are mutually exclusive, *i.e.,* it is not possible to specify a profile and a BC for the same policy. However, a BC can be specified instead of a profile.

- **Profile**: determines how the PDP selects a BC from the BC pool. Thus, it is possible to select a BC with the highest transaction rate (*i.e.,* the `fastest`), or the `cheapest` BC, *i.e.,* the BC with the lowest transaction cost.
- **BC**: determines a particular BC available in the pool to store incoming transactions.

Filters determine the set of options restricting the selection of BCs. The following filters are available:

- **Deployment Type**: specifies the deployment type of BCs, for example `private` or `public` being these values mutually exclusive.
- **Transaction Rate**: determines a transaction rate equal or higher than a threshold.
- **Transaction Costs**: configures a transaction cost equal or less than a threshold defined in a external configuration. This field is only applicable to the `fastest` profile.
- **Maturity**: specifies two fields, which are `stable` and `popular`, in which these are not mutually exclusive.
- **White-list and Black-list**: determines strictly the use or non-use of certain BCs.

Modifiers are a set of options that are applied in the data or in the transaction by the component which enforces the BC transaction. The following modifiers are available.

- **Redundancy**: configures that the transaction data is stored on a BC and in a traditional database, *e.g.,* PostgreSQL.
- **Encryption**: determines whether the data should be encrypted before including the transaction.
- **Splitting**: specifies that the transaction should be sent to all selected BCs.

C. Intent Refinement

The *refinement* process requires the *translation* of the intent and its *validation*. The *translation* process involves the parsing of the intent, using the state machine depicted in Figure 2 based on the grammar defined in Section III-A.

The parsing of the CNL is performed in a Deterministic Finite Automaton (DFA), being responsible for parsing and validating intents. Figure 2 illustrates a high-level overview of the DFA. It consists of composite states, which are illustrated in dedicated state charts, but omitted in this paper due to space constrains and presentation aspects. The states of the parser state machine are named according to the tokens they expect. There are optional states, *e.g.,* the *timeframe* state, that is not mandatory. Also, there are states which can be visited repeatedly, *e.g.,* the *users* and the *modifiers* states. Moreover, there are two accepting states, *i.e.,* the *policy* and the *default policy* states. These correspond to valid intent specifications that can be translated to low-level policies. Finally, there is an error state which is not illustrated in the state chart. However, each state can lead to the error state, if it encounters an invalid event. The error state does not define any outgoing transitions. Thus, the error state acts as a sink state that ignores all further tokens.

1) Translation: After an intent has been parsed, it can be translated into low-level policies. The translator component accepts an intent and processes the parsed options into corresponding options of low-level policies. Intents support multiple users, while low-level policies always correspond to a single user. Therefore, a single intent can be translated into a set of low-level policies, *i.e.,* one low-level policy for each user specified in the intent. Furthermore, the translator component validates the intent options. For example, it excludes all the blacklisted BCs from the pool. Depending on the specified options, it is possible that the resulting pool is empty, *i.e.,* contains no BCs. In this case, the translation ends with a validation error to avoid the output of low-level policies containing an empty BC pool.

2) Validation: The parser and translator components validate the intent based on the specified options. However, not every state of the parser performs validation. Some states perform basic validation, *e.g.,* check whether an option is within a given range or a member of an `enum` definition.

Other states perform more complex validation, these complex validations distinguish between *exclusions* and *conflicts*.

Exclusions correspond to options that have no effect in the current configuration. For example, the `fast` option is redundant if the `fastest` selection strategy has been specified. If an exclusion is encountered, the underlying option is simply ignored. The corresponding policy is still valid and the parsing process continues. *Conflicts* are combinations of options that are mutually exclusive. For example, the `public` and `private` filter options conflict with each other. There are no BCs that are public and private at the same time. Thus, it would result in an empty BC pool of the corresponding policy. Whenever a conflict is detected, the parser transitions to the error state and the underlying policy is invalid.

D. Low-Level Selection Policy

The result of the refinement process is a low-level BC selection than can be utilized by frameworks that support multiple BC and clients, such as [14]. Listing 1 presents such a policy in JSON format. This policy was refined from the intent "*for clientX in the day select the fastest public and stable blockchain except Bitcoin with encryption until the daily costs reach 20*." As it can be seen, the policy contains several details that are not present in the intent, such as start and end time frames, BC pool (not including Bitcoin), transaction rate, block time, and the *encryption* flag set to true.

```
{
  ”user”: ”clientX”,
  ”CostProfile”: ”performance”,
  ”timeframe_start”: ”06:00”,
  ”timeframe_end”: ”18:00”,
  ”interval”: ”daily”,
  ”currency”: ”usd”,
  ”threshold”: ”20.0”,
  ”split_txs”: ”False”,
  ”blockchain_pool”: ”{’hyperledger’,’stellar’,’
      multichain’,’eos,’iota’,’ethereum’}”,
  ”blockchain_type”: ”public”,
  ”min_tx_rate”: ”4”,
  ”max_block_time”: ”600”,
  ”min_data_size”: ”20”,
  ”max_tx_cost”: ”0.0”,
  ”min_popularity”: ”0.0”,
  ”min_stability”: ”0.5”,
  ”turing_complete”: ”False”,
  ”encryption”: ”True”,
  ”redundancy”: ”False”
}
```

Listing 1: Refined Low-Level BC Selection Policy

E. Prototype Implementation

The prototype was implemented in Python. Python is an object-oriented, interpreted, dynamically, and strongly typed language, with a rich ecosystem of third-party packages available. Python is well-suited to implement the system, because it can be integrated with lightweight Web Server Gateway Interface (WSGI) web application framework, such as [19], to provide user interaction without the need for implementing complex bindings or exposing a RESTful API.

The prototype implementation comprises a small number of dependencies. The parser component relies on the Natural Language Toolkit (`NLTK`) for tokenizing intents. The repository is implemented using `sqlalchemy` as an Object-Relational Mapper (ORM). An ORM abstracts the database interactions and is agnostic of the SQL-dialect. Therefore, the repository can easily be extracted and even the underlying database can be switched without affecting the rest of the prototype. Currently, the database is based on PostgreSQL, but it could be replaced with, *e.g.,* MariaDB (MySQL) by simply replacing the database driver. This is possible, because the prototype does not rely on Postgres-specific features. `Psycopg2` is a database driver for Postgres. This driver would have to be replaced when switching to a different database, such as MariaDB. Finally, to create policies from the refined intents in the policy-based frameworks, the `requests` package is required to interact with RESTful APIs via POST/GET requests. The prototype implementation can be found online [21].

IV. Evaluation

To evaluate the feasibility of the proposed BC selection intent grammar and refinement approach, two distinct evaluations were performed. The first evaluation focused on determining the performance of the refinement approach and is described in Section IV-A, along with a comparison with a regex-based approach, presented in Section IV-B. The second evaluation, described in Section IV-C, concerned a survey in the form of a questionnaire on the intuitiveness of the intent CNL.

A. Performance Evaluation

The tests were conducted on the same bare-metal machine, an Intel i5-3570K CPU @ 3.40 GHz with 16 GB of RAM, running Arch Linux kernel 5.4.1. As the state machine was implemented in Python, which is a garbage-collected programming language, the tests were performed with the garbage-collection enabled deliberately. Enabling the collector assures that the measurements are accurate and relate to real-world usage. Disabling the garbage collection and switching to manual garbage collection showed to improve the stability of the results. However, the results not presented a significant variance. Thus, key findings were not affected for automatic and manual garbage collection.

In the first part of the performance evaluation, the refinement process was evaluated. The measurements are performed over 1000 iterations. In each iteration a random intent from Table III is selected and refined. Then, the time measurements are collected and statistics, such as the mean and the standard deviation, are computed for each group and variation combination separately. Finally, a bar chart is plotted using the mean values from the measurements for the evaluation. The error bars depicted in the graphs represent the 95% confidence interval. It is important to note, that the calculation of confidence interval assumes a normal distribution of the measurements.

TABLE III: Intents Employed in the Evaluation

Variation	Complexity	Intent
Single User	Simple	"for *clientA* select the *cheapest* blockchain until the *daily* costs reach *10*"
	Intermediate	"for *clientA* select the *cheapest* blockchain until the *daily* costs reach *CHF 10*"
	Complex	"for *clientA* select the *cheapest public*, *stable* and *popular* blockchain *except Bitcoin* and *Ethereum* with *splitting*, *redundancy* and *encryption* until the *daily* costs reach *CHF 10*"
Multi User	Simple	"for *clientX*, *clientY* and *clientZ* select the *cheapest* blockchain until the *daily* costs reach *10*"
	Intermediate	"for *clientX*, *clientY* and *clientZ* the *cheapest* blockchain until the *daily* costs reach *CHF 10*"
	Complex	"for *clientX*, *clientY* and *clientZ* select the *cheapest public*, *stable* and *popular* blockchain *except Bitcoin* and *Ethereum* with *splitting*, *redundancy* and *encryption* until the *daily* costs reach *CHF 10*"

Figure 3 depicts the average duration of the refinement, where the *x*-axis represents the intent complexity and the *y*-axis represents mean refinement duration in milliseconds. The results for the intermediate and complex intents are similar, indicating that the number of parameters does not affect the performance of the refinement. There is a measurable difference in performance when comparing the simple intents with the intermediate or complex ones, indicating that the specification of a non-default currency affects the performance of the refinement. The duration to refine intermediate and complex intents increases compared to simple ones. Finally, there is a difference in performance for all categories of intents between the single and multi-user variations, indicating that the specification of multiple users affects the performance of the refinement. Specifically, the multi-user variations have a higher duration for all categories, than the single-user variations.

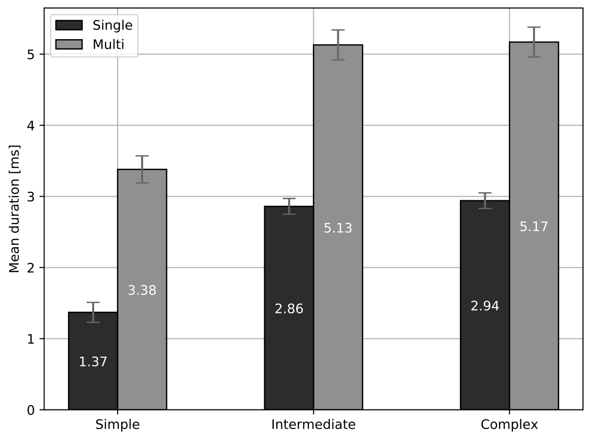

Fig. 3: Average Duration of Refinement for 1000 Iterations

In the second part of the performance evaluation, the refinement benchmark was repeated for a different number of intents, *i.e.,* different number of iterations, ranging from 10 iterations to 10 000 iterations. These tests were executed for all intent complexities and variations presented in Table III.

Figure 4 depicts the total duration of the refinement, where the *x*-axis represents total number of intents refined and the *y*-axis represents the total refinement duration in seconds. The refinement performance results show a linear relationship between the number of intents and the total duration for the refinement. Increasing the number of intents increases the total duration of the refinement process. The duration of the refinement of simple intents in the single-user variation increases the slowest – the duration of the refinement of intermediate and complex intents in the single-user variation increase slightly faster. The duration of the refinement of simple intents in the multi-user variation increases similar to intermediate and complex intents in the single-user variation. The duration of the refinement of intermediate and complex intents in the multi-user variation increases the fastest.

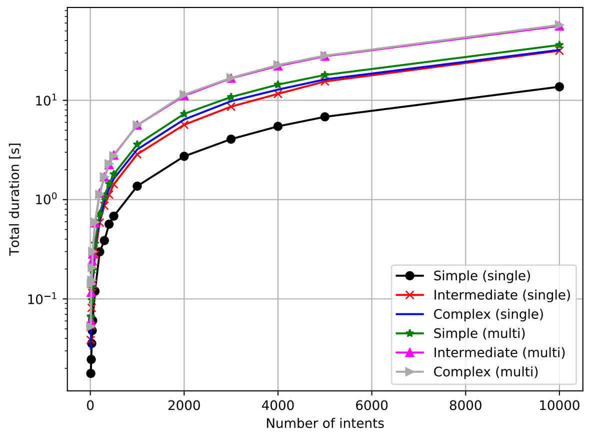

Fig. 4: Total Duration of the Refinement Process

B. Comparison with a Regex-based Approach

Furthermore, for comparison and testing purposes, the intent was parsed and refined using a regex approach (*cf.* Listing 2). Figure 5 depicts the average duration for the refinement of the intents using the regex approach. The *x*-axis of the graph represents the intent complexity and the *y*-axis represents mean refinement duration in milliseconds. Even thought that this approach is faster than the proposed state machine, the regex approach does not allow a granular validation of the intent, *i.e.,* the approach either accepts the intent or does not. In contrast, the state machine allows to exactly pinpoint where the error occurred during the parsing of the intent, providing a feedback to the user.

Moreover, the regex approach is more complex to maintain and is not flexible and extensible as the state machine approach. Thus, the complexity that the regex approach introduces does not justifies its employment solely based on the performance gains.

```
^for (?P<users>[\w ,]+?) (?:in the (?P<timeframe>\w
+) )? (?:select the (?P<profile>\w+)(?P<filters>[\w
,]+)? blockchain(?: (?:from (?P<whitelist>[\w ,]+)|
except (?P<blacklist>[\w ,]+)))?|select (?P<
blockchain>\w+)) (?:with (?P<modifiers>[\w ,]+) )?
(?:until the (?P<interval>\w+) costs reach (?P<
currency>\w+ )?(?P<threshold>[0-9.]+)|as default)$
```

Listing 2: Regular Expression for Intent Parsing

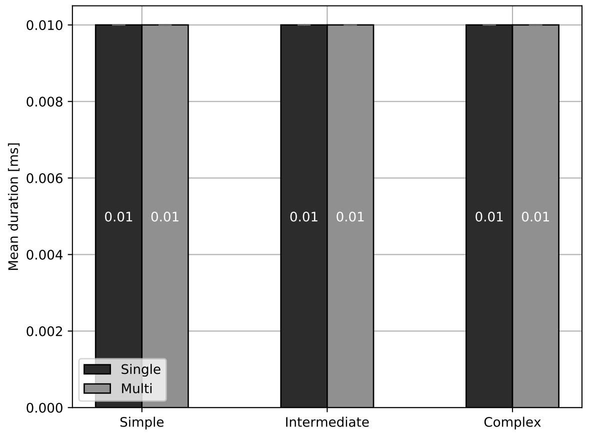

Fig. 5: Average Duration of the Regex-based Refinement Process

C. CNL Intuitiveness Evaluation

The survey involved 19 participants from computer science-related or business and finance-related areas. The period of the survey was from November 1st 2019 to November 30th 2019. The full questionnaire is available online [22].

Respondents were categorized into technical and non-technical individuals based on their BC knowledge. Figure 6 depicts the categories of respondents based on their level of technical knowledge, and background respectively. The 13 respondents who selected the answer "*I know technical details, such as different consensus mechanisms, address formats, node types (e.g., miner and peer), and I have used more than one blockchain implementation*" are considered technical. The remaining 6 respondents that selected a different answer are considered non-technical. Thus, 68% respondents are considered to have technical knowledge of BC, and 32% respondents are considered to have a non-technical understanding of BC. Similarly, 12 respondents selected either "Computer Science" or "Information Systems" as background; thus, a 63% of the respondents have computer-science-related background. The remaining 7 respondents selected either "Banking and Finance" or "Business Administration" as background; therefore, 37% have a business-related background.

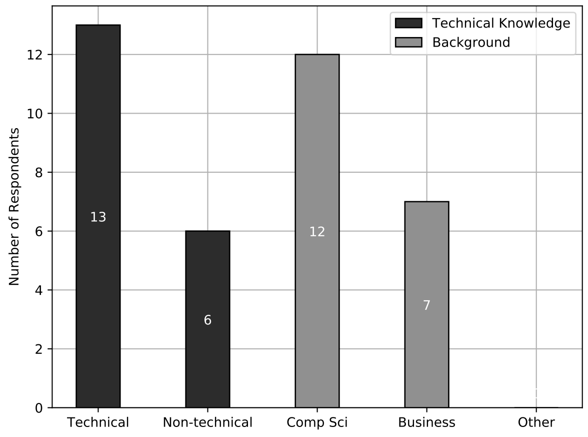

Fig. 6: Respondents Groups

Two Evaluation Questions (EQ) were designed to assess the perceived intuitiveness of the CNL and to compare it with a pseudo-code representation. Based on the results, the Hypothesis (H) formulated to these EQs, were either proved to hold or not. These EQs and their Hs are the following:

EQ1 Do respondents perceive complex queries (in a particular representation) as less intuitive than simple queries, *i.e.,* does the complexity of the query affect the intuitiveness of the query?

H1 *The complexity of the query directly impacts on its intuitiveness.*

EQ2 Which of the two presented representations (*i.e.,* pseudo-code and CNL representations) do respondents perceive as more intuitive?

H2.1 *Non-technical respondents find the CNL representation more intuitive.*

H2.2 *In contrast, technical respondents find the low-level policy more (or equally) as intuitive as the CNL representation.*

To evaluate these Hs, two questions with different representations of BC selection intents were presented. The first question contained only intents represented as pseudo-code (*cf.* Listing 3), and the second question contained only intents formulated in NL following the example presented in Listing 4. The examples used in both questions correspond to the same selection criteria. Each question provided examples A, B, and C, which were categorized based on their complexity. Examples A and B specify only required parameters and are considered *simple* intents. Example C specifies the required parameters and optional parameters as well; thus, considered *complex*, as Table III presents. Respondents were asked to rate these examples in their different representations based on their intuitive understanding. A respondent could select a score, based on the Likert scale, from 1 to 5 or select N/A for not applicable, where 1 means "not at all intuitive", 2 means "not very intuitive', 3 means "neutral", 4 means "somewhat intuitive', and 5 means "very intuitive".

It should be noted that, for the representations in the graphs, the values started as 0 for "not at all intuitive" and 4 for "very intuitive". The results for EQ1 showed that, on average, simple examples were rated as more intuitive than complex ones with a mean score of 3.08 and 2.56, respectively. Therefore, *H1* holds, and the complexity in fact influences the perceived intuitiveness, meaning that the complexity of intents does affect the perceived comprehension of the intent. Simple intents with fewer parameters are perceived as more intuitive than complex ones with many different parameters.

```
{
  "clients": ["B"],
  "profile": "fastest",
  "filters": ["private"],
  "interval": "daily",
  "currency": "CHF",
  "threshold": 30
}
```

Listing 3: Pseudo-code representation

```
For client B
select the fastest private blockchain
until the daily costs reach CHF 30.
```

Listing 4: Natural Language Representation

Furthermore, Figure 7 illustrates the average scores of the pseudo-code and natural language representations based on the answers for EQ2. It can be observed that, on average, the natural language representation was perceived as more intuitive than the pseudo-code representation for all categories (*i.e.,* Technical and Non-Technical). Therefore, *H2.1* does in fact hold, because non-technical individuals perceive the natural language representation as more intuitive. However, the results also show that even technical individuals perceive the natural language representation as more intuitive which invalidates *H2.2*. Therefore, an intent specified in natural language instead of low level configurations such as pseudo-code benefits both technical and non-technical individuals.

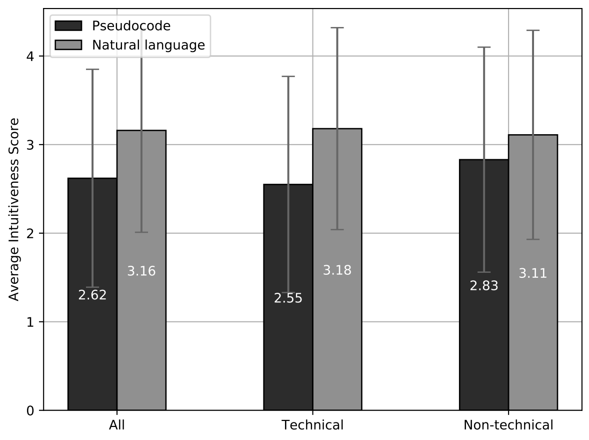

Fig. 7: Average Intuitiveness Score of Different Representations by Technical Knowledge

V. Summary and Future Work

This paper proposed a Controlled Natural Language (CNL) grammar, based on English, to allow non-technical users to specify their requirements regarding Blockchain (BC) selection in the form of intent. Intents are abstract high-level policies employed to abstract technical details from the BC implementations and remove the need for technical knowledge to select the most suitable BC. Moreover, in order to refine these intents to low-level policies, a state machine-based refinement approach was proposed. The approach translates and validates an intent to a low-level policy. Due to the approach's design being decoupled from a single framework, it can be used by a policy-based BC selection framework, such as [14], or as a standalone application to refine abstract high-level requirements to low-level technical BC details, *e.g.,* block time, supported data size in a transaction, transaction rate and costs, and BC type.

The performance evaluation of the refinement approach revealed that its performance is affected by the specification of a non-default currency and the number of users for which intent is specified. However, it is not affected by the number of optional parameters that are specified. The specification of a non-default currency introduces database access during the translation to retrieve the corresponding exchange rate, which introduces a performance overhead. Similarly, the specification of multiple users decreases the performance of the parsing. Moreover, the performance evaluation showed that the duration and the number of intents have a linear relationship. Therefore, refining more than one intent takes linearly more time than parsing a single intent. In addition, an alternative implementation of the parsing based on regular expressions was implemented and exhibited better performance. However, such an approach is more difficult to maintain and does not allow a granular intent validation. Thus, not being suitable for the solution, where error feedback to users is essential.

Furthermore, a survey concerning the intuitiveness of the intent CNL was conducted during a period of one month and counted with the participation of 19 respondents. The results of the survey indicate that both technical and non-technical individuals perceive intents specified in natural language as more intuitive than the same intent specified in pseudo-code. Therefore, even technical individuals benefit from an abstraction layer as provided by the CNL.

Future work regarding the present approach include, *(i)* integration with a policy-based BC selection framework, *(ii)* definition of an approach to quantify the maturity and stability of BCs, *(iii)* research on machine learning and artificial intelligence techniques to aid the refinement of the intent, *(iv)* consider user feedback and revisit the CNL grammar if required, and *(v)* further improvements on the approach.

References

[1] M. H. Behringer, M. Pritikin, S. Bjarnason, A. Clemm, B. E. Carpenter, S. Jiang, and L. Ciavaglia, "RFC 7575: Autonomic Networking: Definitions and Design Goals," 2015, https://tools.ietf.org/html/rfc7575, Last visit December 13, 2019.

[2] T. Bocek, B. B. Rodrigues, T. Strasser, and B. Stiller, "Blockchains Everywhere - a Use-Case of Blockchains in the Pharma Supply-Chain," in *IFIP/IEEE Symposium on Integrated Network and Service Management (IM 2017)*, Lisbon, Portugal, May 2017, pp. 772–777.

[3] W. Chao and S. Horiuchi, "Intent-based Cloud Service Management," in *Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN 2018)*, Paris, France, February 2018, pp. 1–5.

[4] A. Clemm, L. Ciavaglia, L. Granville, and J. Tantsura, "Intent-Based Networking - Concepts and Overview," Work in Progress, https://www.ietf.org/id/draft-clemm-nmrg-dist-intent-03.txt, Last visit December 10, 2019.

[5] CoinMarketCap, "CoinMarketCap Market Capitalizations," 2019, https://coinmarketcap.com/, Last visit June 19, 2019.

[6] Y. Elkhatib, G. Coulson, and G. Tyson, "Charting an Intent Driven Network," in *International Conference on Network and Service Management (CNSM 2017)*, Tokyo, Japan, November 2017, pp. 1–5.

[7] Y. Han, J. Li, D. Hoang, J. Yoo, and J. W. Hong, "An Intent-Based Network Virtualization Platform for SDN," in *International Conference on Network and Service Management (CNSM 2016)*, Montreal, Canada, October 2016, pp. 353–358.

[8] A. S. Jacobs, R. J. Pfitscher, R. A. Ferreira, and L. Z. Granville, "Refining Network Intents for Self-Driving Networks," in *Workshop on Self-Driving Networks (SelfDN 2018)*, Budapest, Hungary, August 2018, pp. 15–21.

[9] J. Kang, J. Lee, V. Nagendra, and S. Banerjee, "LMS: Label Management Service for Intent-driven Cloud Management," in *IFIP/IEEE Symposium on Integrated Network and Service Management (IM 2017)*, Lisbon, Portugal, May 2017, pp. 177–185.

[10] E. Kaufmann and A. Bernstein, "Evaluating the Usability of Natural Language Query Languages and Interfaces to Semantic Web Knowledge Bases," in *Journal of Web Semantics*, vol. 8, no. 4, 2010, pp. 377 – 393.

[11] C. Li, O. Havel, P. Martinez-Julia, J. Nobre, and D. Lopez, "Intent Classification," Work in Progress, https://www.ietf.org/id/draft-li-nmrg-intent-classification-02.txt, Last visit December 10, 2019.

[12] C. C. Machado, J. A. Wickboldt, L. Z. Granville, and A. Schaeffer-Filho, "Policy Authoring for Software-Defined Networking Management," in *IFIP/IEEE International Symposium on Integrated Network Management (IM 2015)*, Ottawa, ON, Canada, May 2015, pp. 216–224.

[13] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2009, https://bitcoin.org/bitcoin.pdf, Last visit December 7, 2019.

[14] E. J. Scheid, D. Lacik, B. Rodrigues, and B. Stiller, "PleBeuS: a Policy-based Blockchain Selection Framework," in *IEEE/IFIP Network Operations and Management Symposium (NOMS 2020)*, Budapest, Hungary, April 2020, pp. 1–9., Accepted. To be published.

[15] E. J. Scheid, C. C. Machado, M. F. Franco, R. L. dos Santos, R. P. Pfitscher, A. E. Schaeffer-Filho, and L. Z. Granville, "INSpIRE: Integrated NFV-based Intent Refinement Environment," in *IFIP/IEEE Symposium on Integrated Network and Service Management (IM 2017)*, Lisbon, Portugal, May 2017, pp. 186–194.

[16] R. S. Scowen, "Extended BNF - A Generic Base Standard," in *Software Engineering Standards Symposium (SESS 1993)*, Brighton, UK, August 1993, pp. 25–34.

[17] Q. Sun, W. Liu, and K. Xie, "An Intent-driven Management Framework," Work in Progress, https://www.ietf.org/id/draft-sun-nmrg-intent-framework-00.txt, Last visit December 10, 2019.

[18] T. Szyrkowiec, M. Santuari, M. Chamania, D. Siracusa, A. Autenrieth, V. Lopez, J. Cho, and W. Kellerer, "Automatic Intent-based Secure Service Creation Through a Multilayer SDN Network Orchestration," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 10, no. 4, pp. 289–297, April 2018.

[19] The Pallets Project, "Flask," 2019, https://www.palletsprojects.com/p/flask/, Last visit December 18, 2019.

[20] Y. Tsuzaki and Y. Okabe, "Reactive Configuration Updating for Intent-Based Networking," in *IEEE International Conference on Information Networking (ICOIN 2017)*, Da Nang, Vietnam, January 2017, pp. 97–102.

[21] P. Widmer, E. J. Scheid, and B. Rodrigues, "Intent Refinement Toolkit (IRTK)," 2019, https://gitlab.ifi.uzh.ch/scheid/irtk-code, Last visit December 19, 2019.

[22] P. Widmer, E. J. Scheid, B. Rodrigues, and B. Stiller, "Blockchain Usage Requirements," 2019, https://blockchain.csg.uzh.ch/BlockchainUsageForm/, Last visit December 13, 2019.