RegelSpraak: a CNL for Executable Tax Rules Specification

Mischa Corsius
HAN University of Applied Science
P.O. Box 2217, 6802 CE Arnhem
The Netherlands
mischa.corsius@han.nl

Stijn Hoppenbrouwers
HAN University of Applied Science
P.O. Box 2217, 6802 CE Arnhem
The Netherlands
Radboud University Nijmegen
P.O. Box 9010, 6500 GL, Nijmegen
The Netherlands
stijn.hoppenbrouwers@han.nl

Mariette Lokin
Ministry of Finance
P.O. Box 20201, 2500 EE
The Hague
The Netherlands
m.h.a.f.lokin@minfin.nl

Elian Baars
Dutch Tax Administration
John F. Kennedylaan 8, 7314 PS, Apeldoorn
The Netherlands
ejh.baars@belastingdienst.nl

Gertrude Sangers-Van Cappellen
Dutch Tax Administration
John F. Kennedylaan 8, 7314 PS, Apeldoorn
The Netherlands
gb.sangers-van.cappellen@belastingdienst.nl

Ilona Wilmont
HAN University of Applied Science
P.O. Box 2217, 6802 CE Arnhem
The Netherlands
Radboud University Nijmegen
P.O. Box 9010, 6500 GL, Nijmegen
The Netherlands
ilona.wilmont@han.nl

Abstract

RegelSpraak is a CNL developed at the Dutch Tax Administration (DTA) over the last decade. Keeping up with frequently changing tax rules poses a formidable challenge to the DTA IT department. RegelSpraak is a central asset in ongoing efforts of the DTA to attune their tax IT systems to automatic execution of tax law. RegelSpraak now is part of the operational process of rule specification and execution. In this practice-oriented paper, we present the history of RegelSpraak, its properties and the context of its use, emphasizing its double functionality as a language readable by non-technical tax experts but also directly interpretable in a software generating setup.

1 Introduction

The Dutch Tax Administration (DTA) is responsible for levying and collecting taxes in the Netherlands. This task comprises a diverse range of taxes, concerning both civil and business tax payers. Changes in tax legislation are carried out yearly, following the annual budgeting process. Every change has to be implemented in the relevant IT-applications, updates of instructions, and communications for tax lawyers within the DTA and for tax payers. This calls for a precise and unambiguous representation of the meaning of tax legislation.

Because the DTA's IT-landscape evolved over the years, various developing methods and tools and various programming languages are still used. On the route from law to automated execution, many interpretations and translations of legislation are produced. These translations are created by a variety of units and teams and are used in various ways. Consequently, there is a considerable risk of faults and of fragmentation of knowledge representations.

The teams of tax and IT-specialists involved in the implementation of legislation noted these issues and expressed the wish for a more coherent approach (Ausems, 2009). This wish was supported by the management, in view of their business goals: time savings and reduction of the risk of mistakes through reuse of rules. In an interview with a DTA stakeholder the common goal was formulated as follows: "*We need an approach that allows us to create a reusable specification of rules (Single Point of Definition), based on the text of legislation and policy rules and validated by tax experts, enabling an automatic translation to high power execution code*."

For example, in the future, the online calculations for income tax on the DTA-website and the web-portals for submitting yearly tax returns will use the same reusable components. Satisfaction of this requirement started with introduction of a new way of working, and a specific rule language: ***RegelSpraak***.

In this paper, we describe the history, the context of use, the users, and the main structure of RegelSpraak, finishing with recommendations for further development and improvement. We believe that RegelSpraak, being a fully operational CNL, is an interesting and rather unique artefact, given its double functional load: it is both human-understandable and indirectly, but fully, executable. In the field and tradition of CNLs (Kuhn, 2009) (Schwitter and Fuchs, 1996), this balance has always been a challenge.[1] The DTA has tackled this challenge in practise and we like to demonstrate how this is done. At the end of this paper, we share the insights and lessons learned from this development process for future research and implementation. This paper is practice-oriented rather than academic in nature, nevertheless it can be a notable contribution to the CNL field and community.

2 History of RegelSpraak

In 2007 the data and rule experts from the DTA got together with the founders of RuleXpress (Rulearts, 2021), a dedicated tool for business rule specification and management. A small team started researching how this tool could be applied within the DTA. They started with Semantics of Business Vocabulary and Rules (SBVR) (Object Management Group, 2021) as a basis, but soon discovered that this standard was not fit for purpose because it did not provide a full set of expressions (i.e. it is not really a rule language, as such). They switched to RuleSpeak (Ross, 2006) as a viable alternative, as it works with informally constrained natural language text and is thus suitable for discussion between lawyers and IT-experts (Baars, 2009). RuleSpeak, however, does not impose a complete set of active syntactic constraints. Therefore, it allowed for too much freedom in rule specification, which limited its usefulness. The team started composing specific language patterns, still largely based on RuleSpeak structures, aiming for a more concise and completely constrained syntax. Consequently, a specific pattern schema was developed. The first version of the rule pattern document (Sangers - van Cappellen and Van Kleef, 2010) consisted of keywords for use in rules, and a number of patterns. These were based on RuleSpeak as much as possible.

The first patterns were written to convert existing rules for Tax Pre-Check (Dutch: Fiscale Voorcontrole) to a readable format. In those early days, MS Excel was used to provide a template for describing rules (see Figure 2). In order to properly and consistently convert the Tax Pre-Check rules to RegelSpraak rules, the DTA first went through the various Excel sheets to determine the different variants of the rules. Patterns and conventions were then drawn up for each of these rule variants. Next, all 2300+ rules had to be converted from Excel to RegelSpraak. A partly automated conversion was carried out. The code-like terms were translated into readable terms based on the Excel sheet and each sentence was converted to a rough RegelSpraak sentence. After that, a manual step was needed to make the rules fully compliant with the RegelSpraak format. The capture of the RegelSpraak (meta)rules was done in RuleXpress, the compilation and transformation in ANTLR. Parr (2014) offers a detailed description of the

[1] In Azzopardi et al. (2018) and also in Calafato et al. (2016), the use of CNL's in a tax and financial context in other countries is discussed.

ANTLR-parser. With this setup, the team started to build rules in XSD/XML.
For the construction and development of a grammar file, the ANTLR Works GUI workbench was used. A grammar and parser were iteratively developed for each of the domains. These parsers were used to check if the RegelSpraak specifications were syntactically correct. Once every possible RegelSpraak statement could be parsed, a compiler could be generated. The intention was to demonstrate that compilers could be made for various different programming languages (e.g. SRL, Java, C++). Consequently, this led to a different format of output for each compiler.
The next step in the development of RegelSpraak was the 'Proof of Concept *Health care insurance law'* (Dutch: Zorgverzekeringswet). In this PoC, for the first time, rules were written based on a direct analysis of the legislation. Thus, the natural language rules were not based on existing code rules but on the text of their legal sources. The analysis of the legislation was carried out along a number of questions that had to be answered for every part (article, section, subsection, sentence, formula) in the legislation. These questions were:

- For whom should something be determined? This is the [role] or [object] in the rule pattern.
- What must be determined? This concerns the [element_to_be_determined] in the rule pattern.
- When should this be determined? This is the [reference_date] in the rule pattern.
- Are there limit values that have to be taken into account? These are the [parameters] in the rule pattern.
- Which keywords are used in the text? These are the fixed natural language terms or phrases.

Several PoCs were carried out to test and refine RegelSpraak. In order to enable automated execution, extra constraints – in addition to Dutch grammar – were needed to achieve unambiguous rules. Parallel to these PoCs, the DTA invested in development of a method and tooling for structured analysis of legal texts, to achieve a solid and precise knowledge base underlying the RegelSpraak-rules. This resulted in iKnow Cognitatie (Cognitatie, 2021) Furthermore, tooling for validation and automatic transformation of the RegelSpraak-rules into code was developed, based on the MPS/Jetbrains workbench (Jetbrains, 2021), now known as the Agile Law Execution Factory (ALEF). Thus, a production chain for rule-based IT development was realized, as illustrated in Figure 7.

Today, RegelSpraak is broadly used in IT development at DTA and is subsequently expanded by newly added sections of the law, and by new patterns on the basis of experiences in applying the language. An important factor in the success of RegelSpraak is its deployment in multi-disciplinary teams which are an important driving force behind the ongoing development of RegelSpraak. The DTA has its own in-house training programme and documents new versions of language and tooling in detailed reference manuals. User evaluation was informally done in the past, see Wilmont et al. (2021) for a more formal method of evaluation.

3 Main Features of RegelSpraak

A basic rule pattern applies to all rules. A RegelSpraak rule always has the following format: [RESULT] IF [CONDITION(S)]. A condition compares attributes, which can have boolean or numerical values, or be dates, enumerations or roles. The result is executed as a consequence of the successful evaluation of the conditions. The results and conditions are connected using carefully composed Dutch phrases to maximize the resemblance to a natural sentence.

For legibility's sake, in our examples, (almost) all examples have been translated into English, without showing the original Dutch fragments.

```
IF (is filled in BALANCE_INCOME-COST_OWNER-OCCUPIED_HOME
[0182] OR
is filled in BALANCE_DECUCTIBLE_COST_OWNER-OCCUPIED_HOME
[0173])
THEN FILL BALANCE_INCOME-COST_OWNER-OCCUPIED_HOME [0172]
WITH
(BALANCE_INCOME-COST_OWNER-OCCUPIED_HOME [0182] -/-
BALANCE_DECUCTIBLE_COST_OWNER-OCCUPIED_HOME [0173])
```

Figure 1: Example Rule 1 without metadata

Rule number	Name	Repetition manner	Rule	Repetition	Consult? Use? Applying	Comment
DA 270	Balance_income_minus_cost_owner-occupied_home	Single	IF (is filled in BALANCE_INCOME-COST_OWNER-OCCUPIED_HOME [0182] OR	N	Single	Rule 1224: Balance income cost owner occupied home: This fact is determined in earlier deduction: the sequence is relevant! total deductible cost owner-occupied home: This fact is determined in earlier deduction: the sequence is relevant!
			is filled in BALANCE_DECUCTIBLE_COST_OWNER-OCCUPIED_HOME [0173])	N	Single	
			THEN FILL BALANCE_INCOME-COST_OWNER-OCCUPIED_HOME [0172] WITH	N	Single	
			(BALANCE_INCOME-COST_OWNER-OCCUPIED_HOME [0182] -/-	N	Single	
			BALANCE_DECUCTIBLE_COST_OWNER-OCCUPIED_HOME [0173])	N	Single	

Figure 2: Example Rule 1 with metadata (Excel-file)

The rule in the Excel file (see Figure 2) consists of two different parts, namely an IF and a THEN-part. RuleSpeak prescribes starting with the THEN-part, usually followed by an IF-part.
Rule 1 is a derivation rule, in which two elements are added up to create a third element. The two elements must not always be added up; the rule contains two conditions under which the calculation must be performed. In this case, one or both elements must be completed.

For an adding up of two elements, the wording MUST BE CALCULATED AS ... PLUS … was chosen, because this comes as close as possible to a natural language expression. The element that is derived is called [element_to_be_determined]. For the condition part, where one or both conditions must be true, IF ... SATISFIES AT LEAST ONE OF THE FOLLOWING CONDITIONS is used. Then the two conditions are mentioned, as an enumeration (the lines with indent in the example below). The rule pattern that fits the result part here is called "basic calculation". A basic calculation is one of the basic rule patterns in RegelSpraak.

As stated earlier, a basic rule pattern applies to all rules, i.e. most rules consist of two parts, namely a *result* part and a *condition* part.

The *result* part is the first part of the rule, describing how the variable that has to be determined gets its value. At the moment, specific rule patterns exist for the result part of derivation rules, restriction rules, classification rules, and process rules. The *condition* part is the part of the rule that describes the conditions under which the variable that has to be determined gets its value. The condition part of a rule starts with the word "if" (see Figure 4 and 6).

In RegelSpraak, rule 1 is expressed as follows:

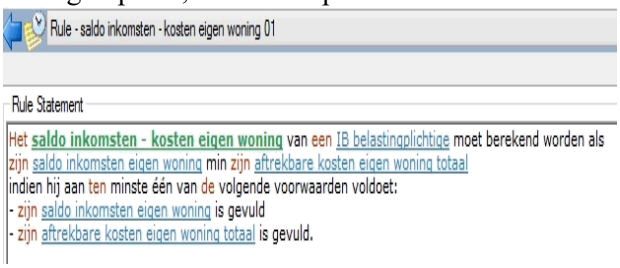

Figure 3: Example of RegelSpraak Rule 1 (Dutch)

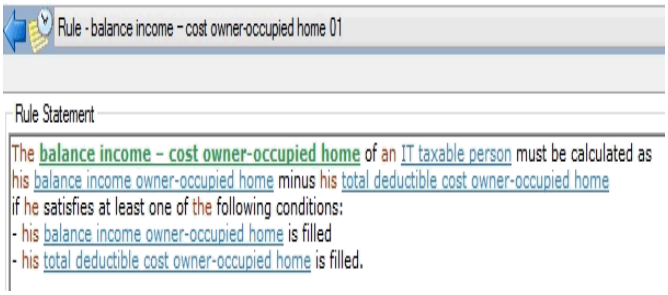

Figure 4: Example of RegelSpraak Rule 1 (English translation)

The basic rule patterns used in this rule are "basic calculation" and "disjunction". A basic calculation consists of a [value] PLUS/MINUS/MULTIPLY/DIVIDE BY [value]. In a basic calculation multiple operators and multiple values may occur. We will illustrate this in the two following examples:
[element_to_be_determined] MUST BE CALCULATED AS [value] PLUS/ MIN/ MULTIPLY/ DIVIDE BY [value]

Figure 5: Example of basic calculation (part of rule 1)

… IF HE /A /THE /EACH PRESENCE OF ITS [object] SATISFIES AT LEAST

ONE/TWO/THREE/... OF THE FOLLOWING CONDITIONS

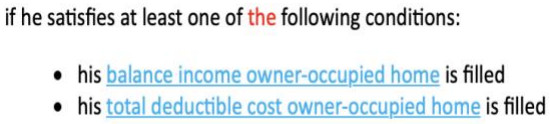

Figure 6: Example of disjunction (part of rule 1)

Rules are often similar in form but different in purpose. For example, there are rules expressing an addition of amounts, or formulas that select the highest amount of two. Rule patterns have been developed to ensure that rules with the same goal are composed in the same way. A rule pattern is a template for concrete rules. By filling this template with the correct elements, an unambiguous rule is created. Each rule must comply with one or more rule patterns.

4 Use and users

RegelSpraak now is a much appreciated and crucial tool in the DTA. The constructive and step-by-step research and development process has paid off, and gradual implementation in the organisation has created substantial commitment and adoption. This allowed the DTA to increase the number of employees working on RegelSpraak in various teams and roles, in a multidisciplinary setting.

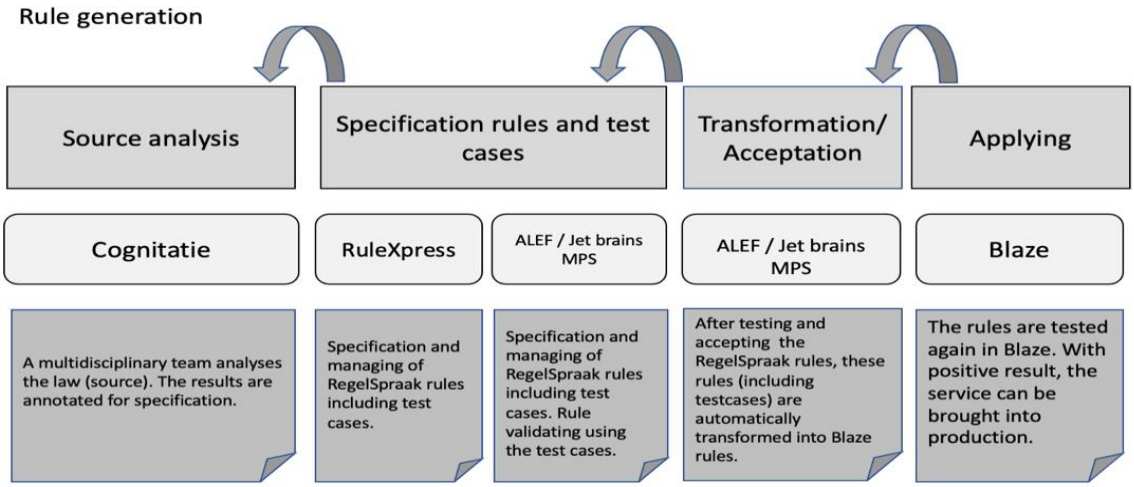

Figure 7: Rule generation

Figure 7 illustrates the rule generation process. It starts when a new or changed piece of legislation is adopted. Tax specialists, rule analysts and IT developers, working in a multidisciplinary team, analyse the piece of legislation in detail and draw up complementary test cases. They make annotations in the legislation using a legal analysing scheme, consisting of predefined labels for legal concepts in the legislation. The annotations are made in *iKnow Cognitatie* (Cognitatie, 2021), a tool especially developed for this. After that, the rule analysts specify rules based on the annotations. In some cases, RuleXpress is used for managing the rules and test cases. It provides repositories to do so. The next step in rule generating is importing this annotated law content into ALEF. In other cases, the rules are specified directly in ALEF. RuleXpress has a more user-friendly interface and better rule management functionality.

The test cases are also added to ALEF, so the rules can be tested. At this point, the tax expert comes in again and validates whether the rules yield the correct test outcomes in the test cases. Finally, the rules are automatically transformed into Blaze-code (Blaze being a high-performance proprietary rule engine (Blaze, 1999)), tested again with the test-cases, and deployed as a webservice.

This process allows for iteration; flaws are now often detected at an early stage in the process. An unambiguous representation, fully in accordance with the tax law, is better achievable. The output of this process is now primarily used as a

component in various back office IT systems, but it is also available for online tax declaration forms and calculation wizards on the website of the DTA.

5 Lessons learned and conclusion

What started off as a small exploration and gradually expanded to PoCs of CNLs in the DTA has now become a robust and scalable infrastructure which to our knowledge has no precedent. The DTA has successfully experimented with RegelSpraak and has gradually become leading within Dutch government in the development and use of CNL. Collaboration with other public organisations is in an exploratory stage, as acknowledgement is growing of the usefulness of shared, reusable rules as a basis for diverse IT applications, and as a central asset in the future of automated law execution.

In the past years, the DTA has documented the RegelSpraak patterns and conventions in manuals, reference guides and workflow-documents. A knowledge community had been created, as well as an in-house training programme. Nevertheless, there is still a lot of tacit knowledge in the heads of the employees, especially with respect to quality aspects of RegelSpraak patterns and rules. The current paper provides a limited overview of what was done so far to make this knowledge explicit. However, further work is needed to validate and improve the framework, patterns, and rule specification procedure. We intend to perform a user research of this, observing and analysing the work of various experts *with* and *on* RegelSpraak.

Lessons learned, for the DTA but also for other organisations implementing CNLs for law execution purposes, are the following:

- Start small: choose a well-scoped and limited part of legislation to start; this will provide an interesting showcase, creating crucial managerial backing.
- Put together a team which comprises the relevant expertise for your organisation.
- Find a sponsor at management level who shows commitment and provides you and your team with the necessary resources (time and tooling), and even more importantly: with space to fail every now and then.
- Invest in education, preferably by training on the job.
- Share knowledge early – pay attention to explicit communication and promotion by means of presentations and publications. This is a good way to share the methods and knowledge within and outside the organisation. The DTA gladly took opportunities to give (invited) presentations, for example at Business Rules Platform Nederland (BRPN) meetings.
- Start testing at once to quickly fix bugs and problems. This saves a considerable amount of money in the overall development project.

6 References

Ausems, A, et al. 2009. Nota Visie Metagegevens & Bedrijfsregels. Version 0.97. Unpublished.

Azzopardi, S, et al. 2018. A Controlled Natural Language for Financial Services Compliance Checking. In *Controlled Natural Language - Proceedings of the Sixth International Workshop*. Kildare, Ireland.

Baars, E. 2009. Nota Eindrapportage: Training Business Rules. Version 1.0. Unpublished.

Blaze software, 1999. Blaze advisory, California. [accessed 2021 May 12]. Retrieved from: https://condor.depaul.edu/jtullis/documents/Blaze_Advisor_Tech.pdf.

Calafato, et al. 2016. A Controlled Natural Language for Tax Fraud Detection. In *Controlled Natural Language – Proceedings of the 5th International Workshop*, Aberdeen, UK.

Cognitatie, the tool that makes your organization compliant and transparent. 2021. PNA Group. [accessed 2021 April 8]. https://www.pna-group.com/kennismanagement-software/iknow-cognitatie/.

Jetbrains. 2021. Jetbrains s.r.o; [accessed 2021 April 8]. https://www.jetbrains.com/mps/.

Kuhn, T. 2009. How to Evaluate Controlled Natural Languages. *In Workshop on Controlled Natural Language,* Marettimo, Italy.

Object Management Group. 2021. The Object Management Group® (OMG®); [accessed 2021 April 8]. https://www.omg.org/spec/SBVR/About-SBVR/.

Parr, T. 2014. ANTLR (Another Tool for Language Recognition). [accessed 2021 April 8]. https://www.antlr.org/.

Ross, R. 2006. The RuleSpeak® Business Rule Notation. In *Business Rules Journal Vol. 7, No. 4.* http://www.brcommunity.com/a2006/b282.html.

Rulearts: Business rules software has a name: RuleXpress. 2021. [accessed 2021 April 8]. http://www.rulearts.com/rulearts-products/rulexpress-business-rules-software/.

Sangers – van Cappellen, G. and Van Kleef, M. 2010. Modelleerpatronen en conventies Specificatieteam; version 0.3. Unpublished.

Schwitter, R. and Fuchs, N.E. (1996). Attempto: From Specifications in Controlled Natural Language towards Executable Specifications. In *EMISA Workshop 'Natürlichsprachlicher Entwurf von Informationssystemen - Grundlagen, Methoden, Werkzeuge, Anwendungen'*, Tutzing.

Wilmont, I., Dulfer, D., Hof, J., Corsius, M., Hoppenbrouwers, S. A quality evaluation framework for a CNL for agile law execution. In *CNL 2021: Seventh International Workshop on Controlled Natural Language*. Amsterdam, The Netherlands.