A Domain Specific Property Language For Fraud Detection To Support Agile Specification Development

Aaron Calafato
Dept. of Computer Science
University of Malta
aaron.calafato.06@um.edu.mt

Christian Colombo
Dept. of Computer Science
University of Malta
christian.colombo@um.edu.mt

Gordon J. Pace
Dept. of Computer Science
University of Malta
gordon.pace@um.edu.mt

Abstract—**Fraud is a major challenge in the taxation area. This is addressed by auditing cases deemed fraudulent by a fraud expert. This selection process currently involves fraud experts defining patterns informally to be developed by an IT team. The code is thereafter tested and refined in an iterative mode until the fraud expert accepts the outcome of the process. This process is very lengthy and prone to human induced bugs. In this paper we present a framework where the fraud expert is empowered to design patterns through a domain-specific language, and feedback will be gathered through an automated process. This process provides the expressivity to define and refine clear and unambiguous rules without the involvement of the software developer.**

I. Introduction

Fraud is a critical aspect in any financial system, including the taxation system. To address this issue, a fraud expert would informally define a tax fraud rule to a software developer. After analysing and understanding the rule, this is coded into an IT system. Before being used in the actual fraud detection, the rule needs to the tested and verified by the fraud expert. Consider the following rule, which might be identified by a fraud expert:

"Load the taxpayer ID for any individual who declared a total income of less than 3000 Euro for any three years of assessment."

If we were to try to get a developer to produce a system which returns the taxpayers matched by the rule, we may run into a number of problems:

1) Typically, when a fraud expert identifies a rule which refers to a particular value e.g. a threshold of 3000 Euros profit, they would generally want to tweak the value so as to improve the effectiveness of the rule.
2) The software might contain bugs due to the software developer's lack of knowledge, especially when dealing with complex taxation concepts.
3) Rules may be misinterpreted, for instance, the "total income" may be understood to be the income from employment rather than the summation of all incomes, such as pensions and bank interests.
4) The process is prone to standard bugs which can occur while implementing any code, due to the manual coding of the system.

The above disadvantages have directed us to tackle the challenges by reducing manual coding in order to diminish bugs and also by defining rules in a more formal manner so as to avoid any ambiguity arising from lack of details. Domain Specific Languages (DSLs) [1] have been found to address both of these problems. DSLs are languages built with only the necessary functionality to address a problem. In this manner, the user can use a focused set of basic concepts to build more complex ones [2]. For example, in taxation, one can combine the notions of "declared income" and "employee" to obtain the notion of "paid salaries".

DSLs are, however, not natural enough to be read from a non-technical user. To address this issue, we have used a Controlled Natural Language (CNL) approach [3] to make the language more accessible to technical laypersons. Just as in the case for general DSLs, CNLs are focused on a particular semantic domain, while being a subset of a natural language, therefore still follow the basic syntax and morphology rules. This makes them more accessible to be used by the common person. Our work uses Grammatical Framework (GF) [4], a framework which allows the generation of CNLs.

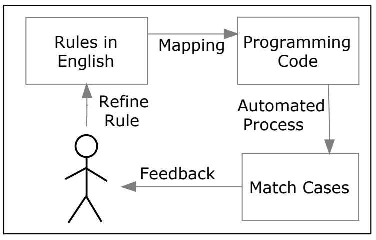

Fig. 1. Automated flow using a Controlled Natural Language

Figure 1 illustrates the proposed flow where the resulting system will allow the fraud expert to define the rules in an English-based language. At this stage, the rules are defined in a clear and structured language. The rules are thereafter translated to the respective programming-code, not readable by a fraud expert but more processable for an IT system. An automated process will then use this generated code to match any cases. To match the cases, our proposed architecture will sequentially process the contents of the tax returns submitted in a taxation system. The processing will involve gathering the fields related to the rule, such as the "total income" and flagging cases matched to the rule.

The returned cases allow the fraud expert to visualise the effect of the rule onto the existing cases. With this approach, the intervention of a software developer is required only at the initial stage to develop the basic concepts. The final result would therefore allow the fraud expert to refine the rule in a “What-if” manner where the feedback would indicate the accuracy of the rule.

II. Language

The two challenges of (i) automation and (ii) allowing the fraud experts to write rules independently, can be achieved through the defined CNL. The benefits are:

- The language is now structured, enforcing that the rules are complete with the necessary information.
- The language is unambiguous, where a term has only one meaning, and a one-to-one mapping exists from the English-based concepts to the programming-based ones.
- The fraud expert is presented with a natural language and not a low level technical language.

To implement the language, the grammar has been divided into two parts: (i) the low-level concepts which are the building blocks and (ii) the high-level forms used to combine these underlying concepts into the tax fraud rules. The low-level concepts are bound to either taxation terms such as “total income” and “individual”, or to standard mathematical terms such as “average” or “three years”. These concepts have been built independently from the high-level structure. For instance, in the above example, the phrase “for any three years of assessment” is built as a low-level component based on smaller and more concise concepts. With this layered approach, intermediate concepts can be introduced, such as “for any three sequential years” uses an additional temporal logic concept.

Defining the grammar in this layered pattern provides a number of advantages:

- Low-level concepts are not tightly bound to the high-level grammar of the language, making the low-level concepts reusable if another high-level concept is introduced.
- Combinators are added to combine the basic concepts into new ones. For instance, a rule may expect a basic condition such as “the bank interests are less than 600 Euro”. With a summing combinator, the field “pensions” may be added to the rule and still preserve the expected form of the condition. The resulting rule “the bank interests plus pensions are less than 600 Euro”. This can be achieved since the summing combinator takes two basic values and returns a new value.
- The high-level grammar is left to control the form of the rule. For instance, the example combines a condition and a number of years for which the condition should be applied. Another rule structure allows the user to aggregate a field for a number of years, such as “an average total income for any three sequential years to be less than 600 Euro”. The two-high level forms common low-level concepts but restrict their composition differently. In this manner, high-level grammar makes the language more controllable since each form is concerned with only one type of rule.

The final outcome from this grammar is a controlled set of concepts which can be used according to the context within the rule. Figure 2 illustrates how CNLs are made clearer with a user interface such as a fridge magnet interface[1], done by only giving the user the possibility to enter correct entries.

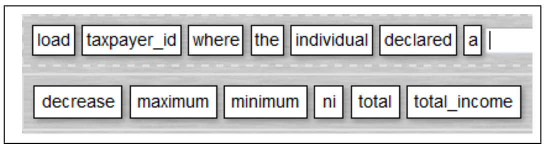

Fig. 2. Grammatical Framework fridge magnet user interface

III. Conclusion

Currently, the process to define tax fraud rules involves continuous human intervention. From the informal definitions to the manual implementation of rules, the current system is lengthy and error prone. This work is directed towards enabling a fraud expert to independently write fraud rules, while being informed with timely feedback on the matched cases for the defined rule. To allow the definition and processing of tax fraud rules, we have proposed the creation of an English based CNL for tax fraud detection. The use of grammars has allowed us to build a well-defined and focused set of concepts which result in unambiguous and clear definitions. With a structured language, the concepts are unambiguously mapped to the respective programming code which is used to match the cases. These cases are then returned to the user to assess the precision of the rule. With this approach, the fraud expert can refine the rule without any involvement from the software developer.

This approach is able to reduce the continuous human intervention in defining rules. The process is therefore made less error prone to human induced bugs and ambiguity in the definitions. Once finalised, the grammar will be evaluated with a number of tax auditors at the Inland Revenue Department to assess its usability and expressivity.

References

[1] M. Fowler, *Domain Specific Languages*, 1st ed. Addison-Wesley Professional, 2010.

[2] S. P. Jones, J. M. Eber, and J. Seward, “Composing contracts: an adventure in financial engineering (functional pearl),” in *ICFP ’00: Proceedings of the fifth ACM SIGPLAN international conference on Functional programming*. New York, NY, USA: ACM, 2000, pp. 280–292. [Online]. Available: http://dx.doi.org/10.1145/351240.351267

[3] T. Kuhn, “A survey and classification of controlled natural languages,” *Computational Linguistics*, vol. 40, no. 1, pp. 121–170, March 2014.

[4] K. Angelov and A. Ranta, “Implementing controlled languages in gf,” in *Proceedings of the 2009 Conference on Controlled Natural Language*, ser. CNL’09. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 82–101. [Online]. Available: http://dl.acm.org/citation.cfm?id=1893475.1893482

[1] A fridge magnet interface provides assistance in writing phrases with a CNL. The interface provides only the words which satisfy the grammar. GF fridge magnet interface: http://cloud.grammaticalframework.org/gfse/