Knowledge Elicitation within the Knowledge Base Paradigm: Disentangling Domain Knowledge from Decision Making in Industrial Applications *

Marjolein Deryck, and Joost Vennekens

Department of Computer Science, KU Leuven
Campus De Nayer, Sint-Katelijne-Waver, Belgium
{marjolein.deryck; joost.vennekens}@kuleuven.be

Abstract. The use of business rules systems to control operational decisions has gained a foothold in the industry. These systems are excellent to define an outcome based on a series of deterministic rules. However, often there is not exactly one outcome, but multiple possible solutions that might be more or less convenient, e.g.; depending on the preference of the user. A constraint based approach that limits the possible solution space is more appropriate for these kind of queries. In my research I use a combination of rule based and constraint based systems in multiple case studies. The overall motivation is to simplify the development of a system, while ensuring a maximal range of functionalities.

1 Introduction and research questions

The output of a company is the result of different kinds of decisions. Take Netflix for example. The fact that you can watch your favorite show on the channel, is the result of strategic, tactical and operational decisions within the company. Netflix strategically decides to work with subscriptions rather than pay-per-view content as their business model. They implement their model with tactical decisions, such as the amount Netflix wants to invest in their own productions. Finally, the decision on the launching date of a new series, is an operational decision. Even though the importance of each individual operational decision might be relatively small, as these kind of decisions are taken frequently and everywhere in companies, they altogether have a considerable impact on the result. Therefore they are the core focus of my research.

Operational decision are typically guided by company rules or applicable legislation. Rules may present themselves in different variants, may contain a lot of detail and exceptions, and tend to evolve over time. A lot of these decisions are still processed manually by domain experts. This often works well, but at

times, decisions might be flawed or incoherent. Moreover, when valued experts leave the company, it puts the company in a risky situation and involves costs.

Sometimes fragments of the decision logic are automated as part of larger digitization projects. This makes sense, as it is a way to ensure that decisions are taken fast and correct. Typically, requirements for this are detailed in bulky documents. Programs need to be tested elaborately and corrected.In classical code, a lot of work is needed to add or change something, even if the needed information is already present in the program.

Declarative expert systems can offer a solution for the last problem. In it, information is gathered in such a way that the system can reason with and act upon it. Although some applications were successful (see e.g.; [1]), the deployment of such an expert system requires a substantial investment.

An important challenge in this respect, concerns the knowledge acquisition effort needed to create a decent knowledge base [2]. Even rather small-scale applications often build on a vast and changing sphere of domain knowledge. Some of this knowledge is implicit, and needs an additional elicitation step. The formalization of this involves a large effort, and is time consuming. It especially requires the precious time of domain experts, who typically have an (over-)loaded schedule.

My research wants to enable more companies to automate their operational decisions, by addressing the challenges related to the implementation of such an automated system. Hence, my main research question is:
How can the knowledge acquisition process from expert's decision making knowledge be improved to overcome the aforementioned challenges?

2 Used technology

In my research, I use and combine two complementary approaches. In the *constraint based* approach, the initial solution space is constrained by limitations, such as physical constraints, cost or material constraints, legal constraints... Solutions that are not excluded by any of these constraints, are valid. The result could be zero, one or multiple valid solutions. The optimal solution can be chosen from these, based on the preference of the user. In the *rules based* approach, deterministic rules with exactly one output are combined together to come to the final solution. The difference between constraints and preferences disappears. E.g.; for the selection of a material to be used in an application, materials M1 and M2 might both be suitable. As M1 is the cheapest, the decision rule will state that M1 needs to be used for this application. The information that M2 would also be possible, disappears. Consequently, if the price of M1 rises and M2 becomes the cheapest, this change would not be reflected in the decision rule.

The IDP reasoning engine that I use in my research, is able to cope with rules and constraints. It is discussed in the next subsections. Decision Model and Notation(DMN) is a business oriented methodology to formalize and implement business rules. It is discussed in subsection 2.2 and further.

2.1 IDP and the KBP

Knowledge Based Paradigm The IDP engine represents an implementation of the Knowledge Based Paradigm (KBP). The KBP advocates a strict separation between domain knowledge and the way this knowledge is used with different inferences [3]. The domain knowledge is declaratively formalized in the knowledge base. This KB contains data and (logical) relations between elements.

Inferences are developed independently of specific knowledge bases. They allow the user to solve a specific problem with the available knowledge. As both the KB and the inference tasks are developed independently, they each demonstrate a high level of flexibility. For the KB this means that it is easy to maintain, because it only contains domain knowledge. For the inferences it means that they can be deployed on multiple KBs. Similarly, multiple inferences can be deployed on one KB, hence leveraging the existing KB to solve questions that were not foreseen previously.

Knowledge Based Systems A KB can be created in different tools, and different solvers can be used to get a solution for a specific problem. For my research I selected the IDP-system. As the research focuses on real life cases, the KB language needs to have a high level of expressivity to make sure that real life situations can be expressed. Many FO-based language exist, that are all able to model more or less complex constructs. The IDP-language that I use in my research, has the particularly interesting possibility to express inductive definitions, as demonstrated in [4]. Also, as the purpose is to demonstrate the use of KBS in practice, an additional advantage of IDP is the online availability of some generic interfaces, that allows to solicit end user feedbac. Finally, enougkh support needs to be available to assist with solving modeling problems. For the IDP system, this support is both available in my own research group, and through close collaboration with the DTAI research group (who developed the IDP system)

IDP IDP (Imperative/Declarative Programming) is the name of both a First Order (FO) Logic based language used to represent domain knowledge, and the reasoning engine in which it is used [5]. An IDP specification consists of three parts, that together form the KB, and represent a stand alone piece of knowledge on the problem domain.

To solve specific problems, multiple inference algorithms can be used. *Model Expansion*, *Propagation*, and *Optimization* are commonly used inferences. With a vocabulary V, and theory T and structure S over V, *Model Expansion* looks for a model. In a model, all elements of V get a value, such that T and S are satisfied. *Propagation* is the derival of the value of some unassigned element of V based on the partial interpretation in S and taking into account T. The result after propagation is a partial model (i.e.; a model that assigns values to some, but not all elements of V) that extends S and satisfies T, and is more specific than the original interpretation S. Whereas model expansion calculates

any total model that satisfies the given constraints, *Optimization* calculates the best model that satisfies the constraints according to the defined term.

Using IDP IDP offers the advantages typical for a Knowledge Based System (KBS), a system that implements the KBP. The maintainability of the KB, and the flexible use of this KB with multiple inference tasks were already mentioned before. A less obvious use of the KB, is to use it as a training tool for new employees or library of decision knowledge. After all, it does contain all relevant information. This multi-deployability increases the return on the heavy investment of analyzing and creating the KB.

Crucial for an operable KBS, is the possibility to express relations and concepts from the real world at a sufficient flexible and fine grained level. This is possible in the IDP language. FO logic uses logical symbols and connectors to express relations between concepts in a semantically unambiguous way, but for people not familiar with the syntax, reading the statements is challenging and proves to be a stumbling stone.

2.2 DMN

Decision taking is a quintessential activity in companies. Using standardized models to reflect them, enables the portability of these models internally or outside the company as communication tool. Simple and readable models, are a vantage in this regard. Since 2015 a uniform standard for decision models is available in the form of the Decision Model and Notation(DMN) standard [6]. As I want to combine the use of business oriented decision models with a FO-based KBS, it is interesting to note that a formal semantic interpretation of a DMN table in FO has been provided by Calvanese [7]. Moreover, the combination of DMN and IDP has been demonstrated in [8].

DMN gives business analysts a tool to separate business processes from the decisions that need to be taken in the process. It favors a separation of concerns between the two areas, which makes both the process and the decision logic more manageable and agile. DMN provides two ways of looking at decisions. First, a Decision Requirements Diagram (DRD) shows an end-to-end view of a decision, from its inputs over intermediate subdecisions to the final decision. In a top-down view, it shows which inputs and intermediate decisions are required to take the top decision. Second, the details of the underlying decision logic. Typically, this logic is represented in a decision table, which maps each set of inputs to the appropriate output. Different kinds of tables are possible and can be distinguished by means of a "hit policy" that defines how different rows of the table interact. These tables can be considered the KB of the decision logic.

Using DMN Together, the use of these two glasses offer big advantages in decision projects. The DRD allows users to get an overview of the domain at hand, without diving into the specifics of a subdecision. It can be used to delineate the borders of an automation project, and make sure that IT and business owners

are on the same page. The individual decision tables are easily understandable with little or no prior training, because they are small, (almost) syntaxless, and declarative. This means that decision tables can be used to discuss the decision logic between different employees from different services. In fact, tables prove to be so easy to use, that business owners are able to maintain the logic without intervention of IT. Hence, business ownership of decisions increase, and business knowledge can be documented better: centralized at one place and potentially continuously updated by business owners. Finally, the decision logic can also be automated.

The use of DMN runs into its limits when it comes to using the information contained in decision tables. Most toolvendors offer one functionality, i.e.; forward chaining starting from all input data, and calculating a single outcome. Reasoning with partial data usually does not act the way humans would. For example, in the OpenRules application, missing data is ignored ('null') and one possible outcome is presented as the only solution. It would make more sense to reason with the information that we do know, to decrease the solution space, and potentially end up with multiple possibilities instead of one solution. The current applications make a strict distinction between input and output data, and reasoning goes from in- to output. Backwards reasoning is not possible.

Also the principle of formatting knowledge in a decision table, restricts its expressivity. Technically, business rules in a table have the same value. In reality, this might not always be the case. Users might want to make a distinction between a physical constraint and a preference. An example of the first is a physical law, e.g.; if temperature is above 200 degrees, material x and y can be used. An example of the latter would be : if temperatures is above 200 degrees, we use material x (e.g.; because x is cheaper than y).

3 Proposed approach and preliminary findings

As the strengths and weaknesses of both approaches are complementary, we combine them to leverage their respective advantages.

3.1 Combining DMN and IDP

In my case studies, DMN is used as a first step to analyze the domain of interest, build a glossary and discover links between concepts. These models are built with the help of domain experts in multiple feedback runs. Because DMN tables are easily readable, they can be validated (and if necessary, corrected or completed) by colleagues that were not involved from the beginning. The result is a first model, with the advantages of clarity and readability.

Subsequently we use the DMN model as an intermediate model to create the IDP specification. Although this may seem unwieldy, the creation of an intermediate model forces the modeler to think trough the domain completely, independently of the final implementation [9].

Alternatively, for single purpose cases, an application solely based on decision tables, might meet all the needs. Multiple tools are available to this end [10].

Once we have created the IDP specification, it is time to think about the possible uses of the information. For the IDP web-IDE, a number of standard inferences are already available [**?**]. There also exist a couple of generic interfaces, that each incorporate a number of these inferences. New inferences can be created or added according to the needs. This step typically requires close collaboration with the domain expert.

3.2 Case studies

Methodology The proposed approach will be applied to five case studies. They cover a wide area of cases, ranging from tax legislation, doctor planning to financial collateral selection. For each of the cases, a template report is filled out, discussing the background of the case, requirements, DMN and IDP models, and difficulties during the modeling and lessons learned. Conclusions from the cases are analyzed and generalized, and additional developments to improve the proposed combination for similar cases are done. The following section gives an example of a case in which both approaches are combined.

Registration rights [11, 12] When purchasing a house in Belgium, some taxes in the form of registration rights need to be paid. The legislation to determine the exact amount was overly complex. To keep up with all the exceptions, a notary's office requested the development of an application that would allow the entry of information during an interview in any order, point to missing parts of information still needed to take a decision, and explain the final result. The system should be able to reason with each additional piece of information that becomes available.

The intermediate DMN model was created with OpenRules. The fully working model was discussed with the notary. His evaluation proved useful in two ways :

- To get feedback on the modeled information : the formalization of the highly complex domain contained some mistakes in the first (and also second) run.
- To understand better which functionalities the notary requires from the tool.

The largest disadvantage of the DMN tool, was the inability to reason with incomplete information. All information needed to be entered upfront before running the tool.

Therefore the DMN model was translated to an IDP specification. As the domain had been analyzed using DMN, this went smooth: the formalisation of the DMN model took several weeks, while the translation to a fully working IDP specification took only 10 person days. The IDP specification was used in the IDP autoconfig interface [13]. This interface offers the inference of propagation, meaning the for each piece of information that is entered, the unalterable consequences are immediately reflected. This proved to be a large step forward to meet the notary's requirements.

What was still missing, is the possibility to see at a glance which questions are still relevant to come to the optimal tax rate, and a way to explain the outcome. To meet these additional requirements, the existing inferences of explanation and relevance were integrated in an advanced version of the interface.

Half way through the project, the legislation was profoundly reformed. The KB was amended to reflect these changes. Even though the change had a significant impact, the modeling effort (after analysis) was limited to a meager half day. This supports the KBP claim that the separation of knowledge and the way it is used indeed advances the maintainability of the KB.

4 Contributions

With my thesis I anticipate to contribute to the current body of knowledge in several ways:

- Give a (partial) answer to the question how and in which circumstances companies adopting DMN can realize the proclaimed benefits. In the product design case, a DMN application sufficed for the selection of standard seals. In the registration rights case the flexible enactment of multiple inferences is needed.
- Make suggestions on how to apply DMN and propose extensions to the standard. The development of DMN+ makes a start with this.
- To demonstrate how to use knowledge bases and different inferences in selected domains.
- To demonstrate the importance of interactivity. The interactive reasoning with partial data turns out to be important in the applications of the case studies.
- To provide general guidelines on how to do this analysis in other domains.

5 Future steps

During the first period of my research, a lot of attention was devoted to the development of solutions for the different case studies. In the following months, the crosswise comparison and abstraction of findings from each case, will lead to general applicable conclusions. The analysis will also point to blind spots. In the following year, additional cases will be executed to address these. After the additional cases, a new round of analysis will allow me to further refine the preliminary conclusions. The DMN+ extension that we started developing recently, will be further developed during the following months: testing the approach, introducing new concepts, implementing full cases. To learn from experts in similar fields, I will undertake an international research visit in the third year of my PhD. During the entire trajectory, the appropriate attention is devoted to the valorization of results: creating prototypes for the case study companies, communicating lessons learned, writing professional and academic papers... Finally, during the last year, some time is blocked to write the actual thesis.

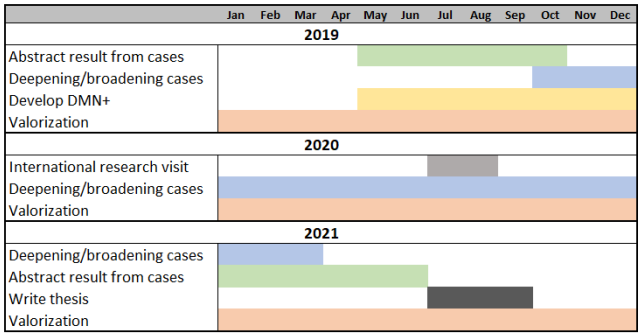

Fig. 1: Planning

References

1. Breuker, J., van de Ven, S., El-Ali, A., Bron, M., Hoekstra, R., Klarman, S., Milošević, U., Wortel, L., Förhécz, A., Estrella, P.: Deliverable n : 4.6 developing HARNESS towards a hybrid architecture for LKIF. (2008)
2. Gaines, B.: Knowledge acquisition: Past, present and future. International Journal of Human-Computer Studies **71** (02 2013) 135156
3. Van Hertum, P., Dasseville, I., Janssens, G., Denecker, M.: The KB paradigm and its application to interactive configuration. Theory and Practice of Logic Programming **17**(1) (2017) 91117
4. Deryck, M., Mitsikas, T., Almpani, S., Stefaneas, P., Frangos, P., Ouranos, I., Boley, H., Vennekens, J.: Aligning, interoperating, and co-executing air traffic control rules across psoa ruleml and idp., Springer (2019)
5. de Cat, B., Bogaerts, B., Bruynooghe, M., Denecker, M.: Predicate logic as a modelling language: The IDP system. CoRR **abs/1401.6312** (2014)
6. OMG: Decision Model and Notation 1.1 (2016)
7. Calvanese, D., Dumas, M., Laurson, Ü., Maggi, F.M., Montali, M., Teinemaa, I.: Semantics and analysis of DMN decision tables. In: International Conference on Business Process Management, Springer (2016) 217–233
8. Dasseville, I., Janssens, L., Janssens, G., Vanthienen, J., Denecker, M.: Combining DMN and the knowledge base paradigm for flexible decision enactment. In: RuleML 2016 Supplementary Proceedings. (2016)
9. Bench-Capon, T.J.M., Coenen, F.P.: Isomorphism and legal knowledge based systems. Artificial Intelligence and Law **1**(1) (1992) 65–86
10. DMCommunity: DMN supporting tools. dmcommunity catalogs, release 2.0.1 of aug 12, 2016. `openjvm.jvmhost.net/DMNtools/`
11. Deryck, M., Devriendt, J., Marynissen, S., Vennekens, J.: Legislation in the knowledge base paradigm: interactive decision enactment for registration duties, IEEE (2019) 174–177
12. Deryck, M., Hasic, F., Vanthienen, J., Vennekens, J.: A case-based inquiry into the decision model and notation (DMN) and the knowledge base (KB) paradigm. In: Rules and Reasoning - Second International Joint Conference, RuleML+RR 2018, Luxembourg, September 18-21, 2018, Proceedings. (2018) 248–263
13. DTAI: Idp. `https://dtail.cs.kuleuven.be/software/idp`