Oussama Chelly and Hendrik Blockeel

10 Industry examples where different AI techniques are combined

In this chapter, we demonstrate how applications can combine different artificial intelligence (AI) techniques from the previous chapters for more advanced applications and how they can cooperate.

The first example shows how an AI based chatbot helps **KBC Group**, a Belgian bank-insurance group, to automate 50 % of the processing of the insurance claims using a chatbot. The second example from the **manufacturing industry** uses a combination of **machine learning** methods and **symbolic AI** techniques to offer a digital engineering assistant that can automatically extract relevant information from engineering drawings and assist the engineers with their choice.

10.1 An AI-enabled chatbot for the Casco[1] insurance industry, an example from KBC Group

Oussama Chelly, Michaël Mariën

Motor-vehicle insurance has been the largest nonlife insurance market over the past decade. In Europe, it accounted for 36 % of the global Property and Casualty (P&C) market with the total motor premium income amounting to €149 bn in 2020.

With over three thousand insurers fiercely competing in the European market, improving the insurance offering for the B2C segment became crucial for every insurer to gain a competitive edge. Consequently, competition was no longer limited to offering better price-to-risk ratios but extended from competition in pricing to competition in the **quality of services** being offered to the end-customer. In this context, **digitally enabled vehicle insurance services** have been gaining more prominence and were estimated to cover as high as 12 % of the vehicle insurance market in Europe as of 2020.

Statistics reflect a **shift in consumer preferences** from the traditional B2C transactions usually carried out in person, over the phone, or by mail to a more digital communication. The change prompted more competition in the race to the digital transformation of B2C insurance policies.

From the consumer point of view, the **insurance claim process** has been—and still is—widely considered to be long and tedious. While the process length and complexity may vary from country to country based on the local regulations, and from insurer to

1 Casco stands for CASualty and Collision (automobile insurance).

https://doi.org/10.1515/9783111426143-010

insurer based on the service license agreements, the driver is rarely satisfied with the quality of service. It should be highlighted that a driver involved in an accident is typically not in the best psychological condition to objectively evaluate the quality of the insurance service. To add more context to the situation in which such claims typically take place, one should rewind back to the moment when the vehicle is involved in a traffic accident. From that moment, the driver is usually under a high amount of stress. Besides any physical injury, post-traumatic effects, or legal consequences, the driver could be faced with the financial implications of healthcare assistance, repairing his vehicle, as well as being deprived of it for an unknown amount of time. That is without mentioning the impact of the accident on his personal and professional plans. Consequently, the claim handling process becomes very sensitive.

In the traditional process displayed in Figure 10.1, the customer notifies the insurance company about the accident by contacting an agent from the company over the phone. In addition to the notification, the customer indirectly expresses his insecurity by inquiring about his current insurance policy. The type of policy, the deductible, and the financial limits are among the most frequently asked questions. Then the customer registers the data about the accident. This data includes the circumstances leading to the accident, information about any other vehicles or persons that were involved, and damage to the parties involved. Once the data is registered, the customer waits while the claim is being processed by the insurance. During this time, the agent has to manually log the data in the company's system and communicate the case to the claim handler. The latter checks for potential fraud, verifies the coverage and liability of the driver, and determines the value of the damage, before issuing the payment to the customer. This process lasts anywhere between a few days and a few weeks. Customer-obsessed insurance companies typically perform this process within 2 to 3 days. While this waiting time is reasonable and competitive, it leaves the customer in a situation of uncertainty, insecurity, and anxiety. In most cases if not all, he is only relieved when the claim is paid out.

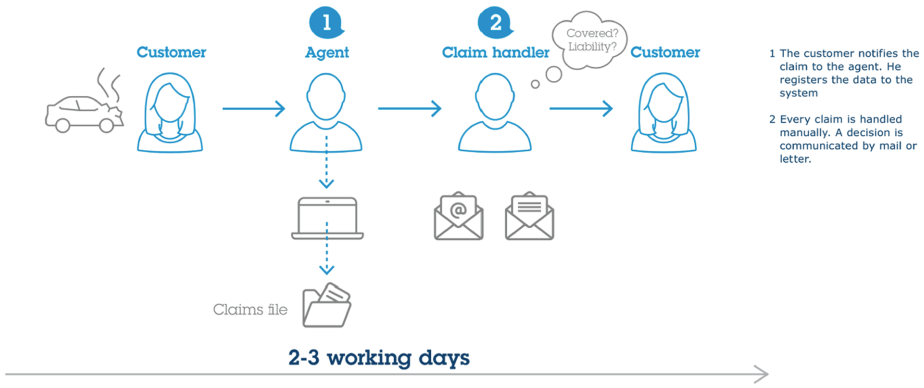

Figure 10.1: The traditional process of claim handling in car insurance.

From the insurers' point of view, the claim processing is a time-consuming and costly endeavor. Even in the most straightforward claim cases, the decision is manually made by a claim handler. Any partial automation of the process would significantly enhance the claim handling capacity of the company, increase consistency of the claim decisions, decrease the costs of handling, and reduce the processing time for both the company, and more importantly, the customer.

Many insurers have invested significantly in the modernization of their services to automate several parts of the process. **KBC Group**, an integrated bank-insurance group from Belgium (see chapter 7), offers a **AI-enabled service** to its customers since November 2018.

In KBC's service, the claim process has been drastically shortened in particular for the straightforward cases (*cf.* Figure 10.2). When a vehicle has an accident, all the driver needs to do is to connect to the insurance app on his phone to start the process. Additional digital channels including the company's website are also an option that the customer can opt for. The process that follows can be divided into three phases: categorizing the claim, assessing the damage, and making a decision.

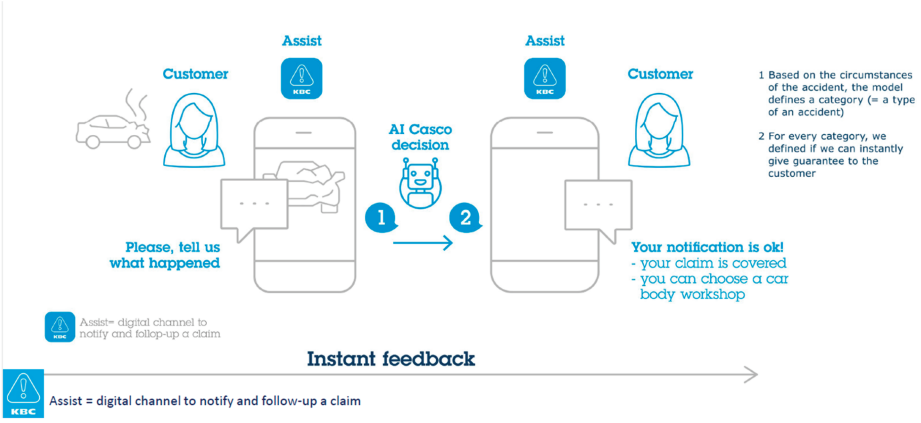

Figure 10.2: The modern claim handling process for car insurance in the KBC Group application.

In the first phase, the customer interacts with a **chatbot**, which plays the role of the company's agent. The chatbot is implemented using **Rasa**, which is an open-source framework for building intelligent chatbots. The chatbot can be decomposed into an **input module**, which uses **natural language understanding** (NLU) algorithms to understand user inputs, and an **output module**, which performs **natural language generation** (NLG) to produce human-like text. The NLU module starts with vectorization to convert the text into a **vector**, then a classification to associate the vector with an **intent**. In parallel, each sentence is tokenized, then chunked, before **named entity recognition** is performed and, therefore, entities are identified. With Intents and Entities identified

from each step in the discussion, business logic is used to allow the chatbot to react to every user input through the output module. This output module relies on a **long-short-term memory** (LSTM) network. As explained in Chapter 7, this type of **recurrent neural network** (RNN) has the ability to maintain a neural representation of the dialog history. Consequently, the context of each sentence in the dialog is inherently maintained.

Besides answering the user's questions on the process and insurance policies, this chatbot captures the story of the accident in text format. The text is then forwarded to two natural language processing (NLP) **classification models** in order to categorize the accident in one of many categories such as "collision with an animal" or "collision with a vehicle." The first of the two models is a **k-nearest neighbors (KNN)** model, which is a well-established classification technique, while the second is a more recent **recurrent neural network** (RNN). The use of two models is motivated by the "four-eye principle" as explained in Figure 10.3.

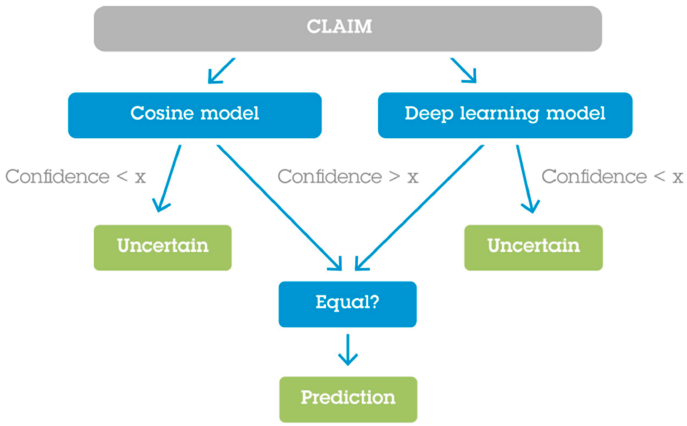

Figure 10.3: Handling the claim uses two models. A decision is only made when both models yield the same category.

In order to use the k-NN model, the customer description of an accident is first transformed into a binary vector indicating the presence of certain predefined key words. Let's assume that our keywords are the following: (car, deer, highway, wall). If the customer says "I was driving on the highway when a deer crossed the road, and I could not avoid it," then the corresponding vector would be $(0, 1, 1, 0)$. The resulting vector is then compared to a set of known vectors extracted from a historical database. The distance metric used to assess similarity is the **cosine similarity**. In other words, the similarity between two vectors is measured by the cosine of the angle separating them. After comparing the vector to those in the historical database, the ***k* most similar vectors** are extracted. The vector is labeled with most represented category in those k vectors.

To use the RNN model, the phrase provided by the driver is first tokenized. After this process, we obtain a sequence of numbers representing the position of each word in the tokens' dictionary. The vector is then fed into a series of **long short-term memory** (LSTM) networks, which can transform the representation of the vector based on the words interdependencies. Therefore, the order of the words plays an important role in the classification. Finally, a dense network is used to label the transformed vector and, therefore, the original corresponding sentence.

Based on the test data, both models agree on a category with a high confidence level in 65.2 % of the cases. On these cases, the **accuracy of the classification** is 99.4 %. In this scenario, the claim is labeled with the detected category and **processed instantly**. In the other scenario where the models yield a low confidence score or disagree on the category, the claim is then **processed manually**.

In the second phase of the claim, the customer submits photos of the accident. The photos are processed using a **convolutional neural network** to identify the type and extent of the damage. This data is then crossed with information on the car to assess the cost of repairs.

In the third and final phase, the insurance app is able to retrieve the customer data and instantly confirm if his insurance policy covers the claimed accident. The chatbot also offers the customer the chance to ask his questions which are then categorized in the same way using an LSTM model to reply with the appropriate answer.

The whole process takes less than a minute before the claim is categorized and resolved. This alleviates the uncertainty on the customer side. It is worthy to mention that during the process the claim data is processed in the background to **detect potential fraud** which was covered in Chapter 7.

In order to improve the quality of its models, KBC implemented a **retraining loop** for the claims where the two models were uncertain or did not agree (Figure 10.4) . In addition to those claims, 10 % of all claims labeled by the two models are evaluated by

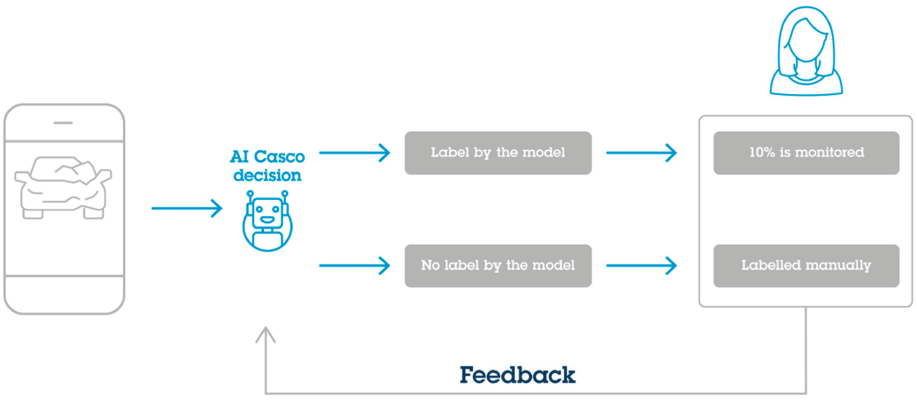

Figure 10.4: The retraining loop in the AI models used for claim categorization.

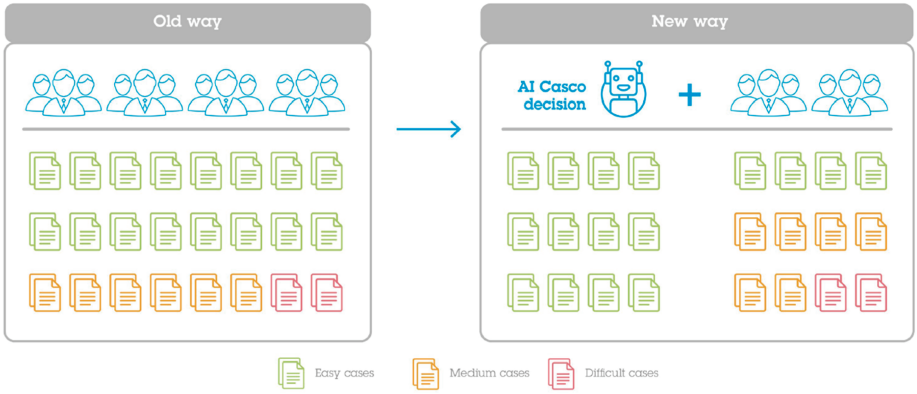

Figure 10.5: The distribution of claims based on their complexity in the traditional and new claim handling process at KBC Group.

human agents to **monitor the quality** of the model. All manual labels are then fed back into the training of the models to improve classification quality.

The outcomes of the new product are overwhelming for KBC. In fact, **50.4 % of the claims** are currently fully automated (*cf.* Figure 10.5). Moreover, by replacing human intervention with AI in the core process of claim handling, the organization has become more cost-efficient and focused on complex claims. With AI processing half of the claims, the complex claim can get more time and attention, which results in better service quality for the insurance customers.

10.2 An automated engineering assistant that uses a mix of learning and reasoning techniques in manufacturing[2]

Hendrik Blockeel, Wannes Meert, Joost Vennekens

This example is taken from a collaboration between **KU Leuven University** and a multinational company active in design and manufacturing. The company produces parts for all kinds of machines. Given a specification of the functionality of some required part, this part needs to be designed and manufactured. The goal of the project was to create an

2 Section 10.2 is based on two scientific papers: (1) Van Daele et al., 2021; (2) Aerts et al., 2022.
The research was supported by Flanders Innovation & Entrepreneurship (VLAIO O&O project 'Digital Engineer'), the Flemish Government ("Onderzoeksprogramma Artificiële Intelligentie Vlaanderen"), and the European Research Council (ERC) (Horizon 2020 research and innovation programme, grant agreement No. 694980, SYNTH: Synthesising Inductive Data Models).

automated engineering assistant using AI technology to help with this. The assistant should allow design engineers to work more efficiently by better **disclosing expertise** already available in the company. Below, we provide more context and give details on the solution that was developed during the project.

10.2.1 The problem setting

The main type of document used by design engineers is the **technical drawing**. Such a drawing typically consists of 2D and 3D visual descriptions of machines or parts, together with annotations such as measurements, a bill of materials, etc. When designing a new object, engineers produce such a technical drawing.

A customer will typically contact the company with a requirements specification for the product they need. This includes its functionality, the conditions under which it will operate (e. g., extreme temperatures), and so on.

Sometimes, a standard solution is already available for what the customer needs, and the sales department can immediately handle the order. When that is not the case, an engineer is faced with the task of **designing a new product**. Often, they can start from a basic type of design that they know was already deployed and evaluated in the field and adapt it to the needs of the customer; more rarely, they need to design something from scratch.

Engineers obviously use their own expertise when designing a product, but they do not have direct access to their (former) colleagues' expertise. The company therefore keeps a database in which earlier designs are stored, so that engineers can tap into it. Being able to find relevant earlier designs that are close to what is needed in a new use case can boost the **engineers' productivity**. Moreover, such a database helps retain to some extent the expertise of retired engineers.

At the start of this project, a large **database with product designs** was available. This database is linked to databases on sales and after-sales that provide additional relevant information (e. g., what kind of unexpected problems were frequently encountered with a given design). Partially because the database spans many years of expertise, different companies, and multiple regions, it is very **heterogenous**: recent technical drawings are typically stored in a digital format, but older ones are simply scans of drawings on paper. This database can be **searched based on keywords**. There is considerable variance in what keywords are used to describe a design: terminology may differ between different company locations and even among engineers at one location; new types of materials become available over the years; insights on what are the most relevant keywords evolve; typing mistakes, etc. Thus, there is **heterogeneity** not only in the designs themselves but also in the **metainformation** about them. This makes it hard to search the database effectively. Looking up information can take a substantial amount of time and effort from the engineer: it may take many attempts before a sufficiently good combination of keywords is found (one that yields a relatively small set

of previous designs that are sufficiently relevant to be useful). Even then, there is no guarantee of completeness: perhaps the best design to start from is not even in this set.

A better way of disclosing the database could significantly increase the efficiency with which engineers can do their job. Hence, **one goal** of the project described here was to build an AI-based system that helps the engineer to find **relevant information** in this database.

By themselves, however, relevant past designs provide only a limited amount of information. For instance, the engineer has no way of knowing whether the past design was actually successful. In addition, the solutions that used to be optimal, perhaps are no longer optimal today (e.g„ because new, superior materials have been invented). Perhaps most importantly, a design drawing tells the engineer which design choices were made, but not ***why*** these choices were made.

The company therefore also had a **second goal**, which was to extract the knowledge of key senior engineers and to explicitly store it in a formal **knowledge base**, such that it will remain available for future generations of engineers. In addition, this knowledge can then be used to provide **flexible and explainable decision support** to the engineers.

10.2.2 How are those problems solved ? A mix of techniques from Chapters 4, 5, and 7

The developed software uses multiple AI technologies to assist the design engineers: it combines **computer vision, inductive logic programming, pattern mining, knowledge representation, logic reasoning, and constraint reasoning** (Chapters 4, 5, and 7).

10.2.2.1 Reading the drawings

Technical drawings consist of a 2D and 3D drawing (the “CAD” drawing) together with annotations including measurements, a list of parts and/or materials, and so on. A lot of relevant information is in the 2D drawing itself. A vision component was developed that can analyze a drawing and extract relevant information from it. This vision component reads the drawing as a bitmap image (so it works as well with scans of designs on paper as with digital drawings). The image is first segmented using standard **computer vision** methods, and segments are then **classified** as “table,” “two-dimensional CAD,” or “irrelevant.” This classification determines the next processing step: table segments are handled differently from CAD segments, whereas irrelevant segments are ignored. The segmentation and classification were found to be 100 % accurate in the available data (which is in line with the fact that line drawings are generally not very hard to segment).

10.2.2.2 Reading the tables

Data in tables are organized partly through annotation (e. g., column or row titles) and partly through positioning (e. g., all cells below some column title belong to that column). Different tables may have a different organization, however, and tables do not always have a simple matrix form (m rows, n columns): cells may span multiple columns or rows, a cell may contain a subtable, etc. The system therefore needs to learn how to parse tables.

An **inductive logic programming** (ILP) approach was used for this. The ILP system takes as input, descriptions of cells (cell text, cell location), relational information derived from this (relative cell positions, neighboring cells, the order in which cells occur), and labels of the cells. It produces as output "mini programs" that state how to derive the label of a cell from the other information. An example of a rule that the system finds is:

author(A):- cell_contains(B, "drawn"), above(B, A),

which states: when the cell above this one contains the text "drawn," this cell contains the name of the author. Figure 10.6 shows another example of a program and how it interprets a table.

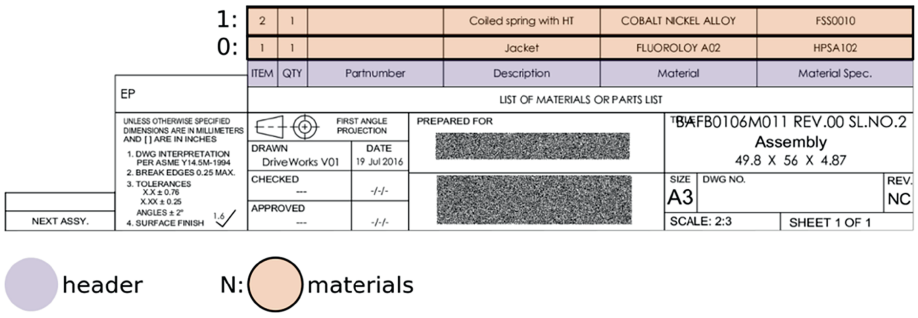

Figure 10.6: Example of a table and a mini-program defining the concepts "bill of materials" and "header." The highlighting shows what the program defines; it is not part of the original drawing. (Figure by Van Daele et al., 2021.)

Rules similar to these were introduced for all labels. Some cells can be classified quite well using such basic rules, others are harder to classify accurately. The overall

quality was substantially improved by introducing a novel element into this ILP approach, called **bootstrapping**. The basic idea is to let classification rules for difficult labels exploit the results of other classification rules for easier labels. A dependency graph is constructed, where labels are ranked according to the accuracy with which they can be predicted and the size of the program predicting it. After a first learning run, simple rules with high accuracy are added to the background knowledge that the ILP system can exploit, then a second learning run is made that can exploit the additional knowledge that has become available in this way. This process is repeated for each consecutive run. One task that benefits from this procedure is to recognize the bill of materials, where first the concept of a header is detected, after which recognizing rows of materials is easy.

10.2.2.3 Reading the CAD drawings

To perform searches based on CAD drawings, a ***meaningful* similarity** measure for such drawings needs to be available. Specifically, two drawings should be considered maximally similar if they represent the same design (making abstraction of rotations, translations, mirror symmetries, etc.) To learn a suitable similarity measure, **self-supervised learning** is used: for each image, 10 more images are constructed with irrelevant variations of the original (e. g., rotating the image); then a "**siamese network**" is trained that for any pair of images should output whether they represent the same design or not (using so-called ***contrastive learning***, where pairs of images derived from the same original are positive pairs, and random samples of image pairs are negatives). The siamese network processes each image in a given pair using the same network (a convolutional neural network with ResNet architecture), then combines the outputs of these networks using a few fully connected layers. In this way, a neural network is trained that can assess to what extent two drawings represent the same design.

10.2.2.4 Identifying relevant designs

With the functionality described in previous sections, it becomes possible to define a measure for the **similarity** between two designs. Each design is first represented using two feature vectors:

– The first relates to the **tables**. As the description of the table resulting from 2.2 uses a logical format, an ILP system called Warmr is used to find **frequent patterns** in the logical description that are likely relevant for determining similarity. A feature is introduced for each relevant pattern. These features form the "tabular" feature vector. For tabular feature vectors, which have binary values, similarity is defined as the proportion of features that have the same value.

– The second feature vector, which relates to the **CAD part**, contains the nodes in the penultimate layer of the **siamese network** that determines whether two drawings represent the same design (these are obviously relevant for the similarity between designs). For these CAD feature vectors, the cosine similarity measure is used.

The overall similarity between two designs is finally defined as the geometric mean of both similarities.

10.2.2.5 The knowledge base

To complement the database of designs, a **knowledge base** was built that captures the knowledge of key domain experts. To construct this knowledge base, several interactive workshops were held, in which experts from different sites worldwide participated, guided by a knowledge engineer. The knowledge base is written in the **FO(.) language**, which is a rich extension of classical **first-order logic** (Chapter 4).

During such workshops, it is important to represent the knowledge in a **formal language** that not only the knowledge engineers but also the domain experts can understand. In this way, the domain experts can immediately check whether the knowledge engineer has correctly understood what they are saying, which greatly reduced the number of mistakes that end up in the knowledge base.

While the FO(.) language is powerful and easy to use for trained experts, it can be challenging for people who first encounter it. The workshops therefore made use of the decision model and notation (DMN) standard, and its extension cDMN. This offers an intuitive table-based representation, which has been specifically developed to be usable by domain experts. An example of a cDMN table is shown in Figure 10.7.

Component Materials				
E*	**Component**	**Component is Used**	**Design Type**	**Material of Component**
1	Body	True	-	M1, M2, M3
2	Spring , Spacer	True	-	M1, M3, M5
3	-	False	-	null
4	Body	True	Closed	Not(M2)

Figure 10.7: A cDMN table that defines which materials can be used for which components in which types of design. For instance, in a closed design type, any material apart from M2 can be used to manufacture the body of the component.

10.2.2.6 Interactive decision support

To build a usable **decision support** tool, it is key that the tool can adapt to the way of working of the engineers, rather than forcing the engineers to adapt their way of working to the tool. A decision support tool was therefore developed using KU Leuven's

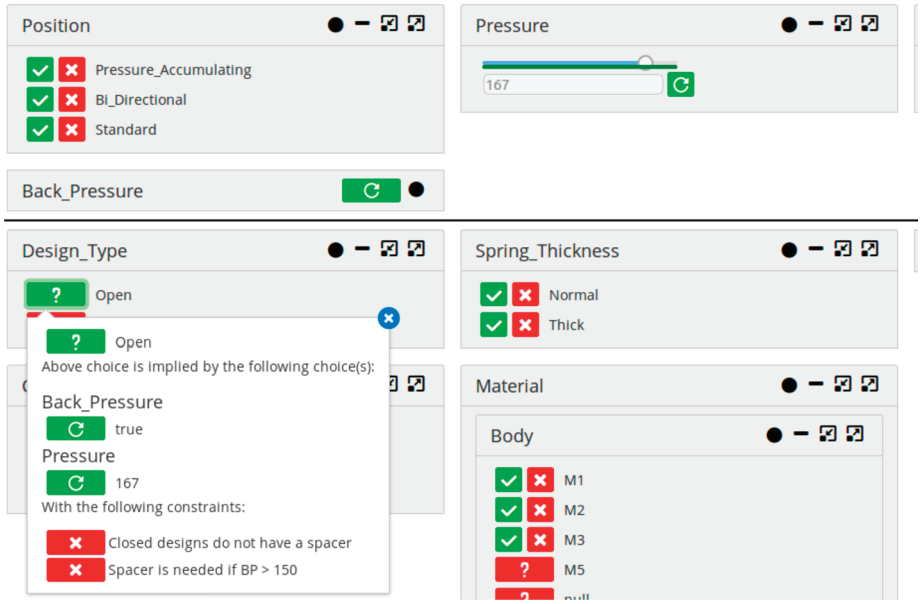

Figure 10.8: The Interactive Consultant providing an explanation for why an open design must be used. This is the case because the engineer has entered the requirements that the design should be able to release back pressure and to cope with pressures up to 167. Other information that the system has already derived is that material M5 cannot be used to manufacture the body of the component.

generic Interactive Consultant interface (idp-z3.be), which is powered by the **IDP-Z3 reasoning system** for FO(.). Figure 10.8 shows a screenshot of this interface.

An important property of this system is that it adheres to the ***knowledge base paradigm***: the knowledge base itself is purely a declarative representation of knowledge (i. e., by itself, it does not *do* anything), to which different logical inference algorithms can then be applied to derive different kinds of conclusions from different input. In this way, the system can give the engineer the freedom to work in whichever way they choose. They can start from the requirements, they can start from a specific design, they can start by choosing materials, etc. Whatever information the engineer chooses to enter, the system will use the knowledge base to derive further conclusions from this. In this way, the engineer and the AI system cooperate to gradually reduce the number of options that remain, until finally a single, complete design remains. Because all the information that the system provides is derived by means of **logic reasoning** from the knowledge base that has been constructed and verified by the domain experts, information coming from this system is at least as reliable as information that an expert would provide themselves. Moreover, all the output is also **explainable**, in the sense that the system can always point to a precise combination of choices made by the engineer and parts of its knowledge base that suffice to reach this output.

At any point during the design process, the engineer can use the same user interface to inspect the database of previous designs. In this case, the current state of the design process is used to return only designs that match this specific context. Moreover, the properties of each design are shown using the same concepts that are used in the configuration interface. This makes it easy for the engineer to spot the key differences between different designs and to copy relevant parts over to their design.

10.2.3 How well does it work?

The time that engineers need to come up with a good design given specifications was substantially reduced. The time spent searching the database to see what already exists, what were the problems with some earlier designs, which designs were successful, and which were not (and under what circumstances) is reduced significantly, by 15–30 minutes per use. This comes on top of the fact that engineers rarely spend more than an hour on this search: if they cannot find anything fast enough, they start designing from scratch, which may take dozens of hours. The AI provides much smarter access to the database, and the interaction allows the engineer to quickly zoom in on the most relevant cases. The integration into a configurator allows the engineer to quickly come up with new variations that meet specific requirements.

Bibliography

Aerts B., Deryck M., and Vennekens J. Knowledge-based decision support for machine component design: A case study, Expert Systems with Applications, 187, 2022.

Van Daele D., Decleyre N., Dubois H., and Meert W. An Automated Engineering Assistant: Learning Parsers for Technical Drawings, In Proceedings of the 33rd Annual Conference on Innovative Applications of Artificial Intelligence, 2021.