Wood selector expert system for practical use cases

Knowledge Technology Practical Project Plan

Nil Stolt Ansó s2705338
Francesco Dal Canton S2935120
Anton Wiehe S2972301

December 2017

1 Problem

There are upwards of 15 000 different types of wood in the world, 2 000 of which have commercial characteristics. From these only a handful are commonly commercialized around the world, each of which is used for different kinds of woodwork projects due to their individual different characteristics.
One of the most obvious characteristics to take into consideration when doing woodwork is hardness. If what you are trying to build needs to be durable, you would want to find a wood that is relatively hard. On the other hand, the malleability of the wood is reduced the harder it is, which might not be something you might look for if your woodwork requires sculpting. This is just one of many characteristics one should take into consideration, with others having to do with response to humidity or the market's ease of supply.
The aim of our program is to use these wood characteristics and properties to guide the user of the program towards the correct wood type for their project. The system will not give one single recommendation, as usually there are many different types of wood that would fit one job. Rather our system presents the woods most suited to the users preferences ordered according to the users needs. For each wood type there is an image of the wood shown, as the aesthetics of the wood might be more important to the user than the exact ranking of the wood.
Our program can be useful for anyone that aims at doing some wood work. This could range from furniture to musical instruments to even saunas. For larger projects, such as building houses or bridges, we advise users to rather consult a human expert on wood.

2 Expert

The expert that we got in contact with is a master carpenter and wood crafter. His name is Timo Stolt. He has worked as a carpenter for decades, many of which have been spent in the field of house construction. He was available for helping in the project extensively and without strong time constraints. He is a direct relative of one of our group members.

2.1 Knowledge elicitation

Our meetings with the expert occurred usually through online video calls as the expert does not live in the country. Usually, prior to a session, we prepared by having a short group session to collect pointer about which questions to ask about the expert's knowledge and opinions towards implementation.
The very first session was had as an unstructured interview through a video call in which we discussed for about 2-3hours about the expert's domain and possibility of implementation. He gave us opinions on which specific domain to focus on, and gave us detailed explanations and analogies on his usual thinking process when coming up with a wood work project. He later that week also sent us a spread sheet with useful data on 40 different types of woods.
Once we had a prototype of the knowledge system, since the expert is a direct relative of one of our group members, we had plenty of opportunities for unstructured interviews and protocol analyses in person during the Christmas holidays when one of our group members went for a visit back home. A great deal of time was spent discussing the expert field, for which the expert showed us a great deal of books on the subject. Information was gathered from these books to further expand the knowledge base. The expert also gave our member a tour in a wood storage house to further emphasize the process of choosing a wood type. It was then that the expert gave us the idea to also include pictures of the individual wood types as pop-ups in the programs as that plays a big role when choosing the appearance that a wood project might have. During, this time period we also had numerous protocol analyses, as the expert was very much at our disposal, to validate the rules we were currently implementing.
Towards the end of the project's making we had two more video calls with the expert for which we had prepared a structured questionnaire for the interview. Most of the questions were meant for validation of our system, although we made room for questions we might have come up during the interview. Overall our intention was to make sure we didn't miss any flaws in our thinking process by asking about pretty much anything. During process of each of these interviews, we also had the expert try navigating through our expert system to make sure he agreed with our implementations. Most importantly, we had the expert take a look at our rules and inference system so as to validate and provide feedback on their implementation.

3 Role of knowledge technology

A lot of knowledge and experience goes into selecting the correct type of wood for a project. To properly guide a user to suitable wood types for his project, the system has to ask specific questions to the user of which it can infer what the characteristics of the wood should be, without confronting the user with all the detailed information of the wood. Our system is equipped with a large database on wood types with many relevant characteristics to extract the knowledge from. This knowledge of the database was gained by consulting our domain expert.
Additionally, our system uses facts and rules that are stored in text files from which it can infer when to exclude woods from the set of suitable woods. This means that our system gains expert knowledge through the database and the rules, which we derived from meetings with the expert, and combines it with knowledge of the project that the user envisions, so as to select favorable wood types.
It is important to note that our knowledge engine is completely separated from the expert knowledge. All expert knowledge is stored in external text files. In theory we could use a database on a completely different topic along with rules, facts and questions associated to that topic and our system would still be able to work with that knowledge base.
Our system implements forward chaining in order to acquire new knowledge. That means that it contains a number of rules, which may be satisfied by determining the truth values that appear in them. The system chooses what rule to fire by determining what fact would maximize the amount of rules that can fire. In other words, the system is at all times trying to maximize the amount of new knowledge.
Gradually, as we were working with the expert, we realized that most times the decisions as to what wood to use in a certain context are guided by extremely straightforward heuristics. That is to say, in some cases an expert will think along the lines of "The user wants to build a boat, and I know for a fact that wood A has always been used for this task.". In other words, sometimes there is not a lot of complex inference going on, but rather, the expert is relying on knowledge that is strongly based on experience and intuition. That is one drawback of our program, in that the knowledge base doesn't involve a particularly complex inference infrastructure. However, it does represent the knowledge of the expert quite accurately. Many sessions with the expert were spent by having the expert give continuous feedback as to how to represent the knowledge more and more realistically.

4 The knowledge models

4.1 The Domain Model

The main goal of our program is to recommend one or several wood types to the user. To filter out irrelevant wood types we need to refer to the properties of

the woods, such as hardness, malleability, resistance to humidity and more. For all the wood types we collected values for these properties and put them in a table. Additionally our expert provided us with columns that indicate for each wood type whether it is usable for a specific endeavor, such as constructing a sauna. For these types of applications, there are only several woods applicable to use. The properties mentioned so far are all used in the inference. For the ordering of the woods we also included an ordering value for each wood type, that gets adjusted as the user answers questions. This ordering value is calculated by multiplying each ordering property by the weight that was assigned to it through the questioning and then we sum the result of every multiplication. The aesthetics of a wood might also be very relevant for the user, therefore we collected pictures for every wood type in our database.
The next important part of our domain model is the Fact class. Each fact can have the truth value True, False or Unknown. Of course at the start every fact has the value Unknown and through the responses of questions by the user True or False get assigned to the facts.
Aside from normal facts, which only have a truth value, there are also filtering facts and ordering facts. When filtering or ordering facts get assigned the truth value True, they get activated. Filtering facts will iterate through the list of available woods in the model and filter out woods. They filter out woods according to the property that the fact was assigned to in the text file. They check if the wood types have that property set to the truth value that is also indicated in the text file, and if they do they mark that wood as unavailable. When an ordering fact is activated, a weight is given to a certain property of wood types. This weight and property were declared in the text file. When the system will later reorder the wood types according to suitability, it will take these adjusted weights into account.
Our domain also comprises rules and questions. For rules we simply store the conclusion and the premises for that conclusion. For each premise and conclusion we take into account whether they were negated. If they were negated as a premise, the rule will only fire if these premises attain the truth value False. If they were negated as a conclusion the rule will set the truth value of the conclusion to false upon firing.
Lastly there are questions in our domain. Each question is a simple yes/no question in the current version of the program, but they can be easily extended into having more options. The question class has an associated question text, as well as for each button it has facts associated to it, that will be set to either True or False, depending on whether they are negated or not.

4.2 The Rule Model

Our rule model is organized such that for each rule a set of true premises can conclude a single conclusion. Each rule is attempted to fire after every answer by the user to check if any new true facts can be concluded. Any fact in a rule can be negated, and negated conclusions can also be made. A majority of rules in our system are simple 1-premise-1-conclusion rules, this is often the case for

rules made to gather basic information from the user, whose conclusions we use in more complex rules later in the system or use to make certain more complex rules not able to fire. An example of this would be our Paint_order rule, which can only ever be fired if the user has answered that he wants to make Furniture elements, Joinery elements, or none of the asked object types. Similarly, if a furniture element's wood is treated internally chemically, the Paint_order concluding rule will never be fired.
There are also facts that, after certain rules fire, are assumed automatically. This is done by single premise rules firing where the premise is the previously-fired rules' conclusion. This has a goal to represent rules which have several conclusions, but that our system does not represent explicitly into one rule, and instead spreads it over several. An example of this would be the ordering of the woods based on Ability to be painted, Price, and Dilation after the user decides he will want to paint the wood. The Painting rule "Paint_order,Paint,Paint_path" (where the first fact is the conclusion followed by two premises) will conclude Paint_order after the user sets Paint to True. In turn, the rules "Price_order,Paint_order" and "DilationPaint_order,Paint_order" will also fire thus leaving us with 3 ordering-Facts to apply to the list of woods. Most of the rules were created hand-in-hand with the knowledge we extracted from the first 2 interviews with the expert. Which assumption the rules make and the technicalities in the order of the firing of the rules, were validated and discussed with the expert in later sessions.

4.3 The Problem Solving Model

A carpenter is someone that undergoes the task of constructing and repairing objects made out of wood. In our case, however, we restricted the task of the expert to the skill of choosing the right wood for a job in order to control the complexity of the system. As such, the problem that we are facing is: how does the user go from their project idea to knowing what woods they can use for it? Initially the user would certainly describe to the expert what project they have in mind. Nevertheless, the user is likely to leave out details that the expert knows to be important, as much as to include details that may be irrelevant to the end decision. That is why the expert has to ask questions to the user which allow them to probe the needs of the user. Here, the expert is not only figuring out what the user has in mind. They are also, at the same time, understanding what the implications of such a project are in terms of the mechanical and chemical properties of the wood, along with requirements that have relatively little to do with the wood's properties, as much as how much the user is willing to spend, for instance.
At this point, the expert has formed an idea about what the user needs, which might be largely different from what the user actually imagined initially. That's because the expert is able to incorporate the idea of the user with their own expertise about what different types of wood are good for, and so the expert can recognize what is actually important and what the implications of certain requirements are. With this understanding, the expert can finally make a de-

cision and give some plausible options to the user, who can then choose what wood to use based on the look of the wood.
This process could be schematically represented as seen in Figure 1.

Figure 1: Schematic representation of the problem solving model.

Our goal was to model this process loyally through our program. When the user starts our program, they are immediately asked a dynamic line of questions, which are probing the user's needs and extracting information about what the user wants to do. At the same time during this questioning process, the program is performing inference in order to reason about the implications of the user's answers, and is asking questions that are appropriate to the knowledge that is still needed to refine the decision.
Throughout this process, the program is collecting information and reasoning in order to properly understand what the user's requirements are, by using the knowledge base that it possesses. At the end of the process, the program is able to give a final recommendation, which is given in the form of a list of woods along with how suitable they are for the project.

4.4 The Inference Engine

As we build our system from scratch we had many options for our inference engine. We had several ideas, such as choosing a question that leads to about half of the available woods being filtered out, to minimize the number of questions we need to ask. Another idea was to look at the best fitting wood at each moment and then look through the database to see if a question can be posed that would filter that wood out. In the end we decided to choose another approach: our inference engine aims at firing as many rules as quickly as possible. We chose this approach because it seeks to maximize the inferred knowledge of the database, so our program can be seen as an artificial expert that wants to know as much as possible, as quickly as possible about the user to give the best possible recommendation.
When looking for a next question to ask for, the inference engine starts by looking at the available rules. It goes through all the rules and counts the unknown

premises in the rule that have questions attached to them. The engine does not check rules that have been fired or that can never fire in the future (because some truth values in the premises are already set in a way to make the conclusion unavailable). Then the system chooses all the rules that have a minimum number of unknown premises in them and looks through the facts within them. It then counts how often each fact appears in this set of rules with minimum unknown premises. The fact that appears most is then selected as the fact of which the system wants to know the truth value. Then a question that can change the truth value of that fact is posed to the user.
It is important to note that the engine prioritizes firing rules as quickly as possible. This is because every conclusion of a rule might activate a filtering or reordering procedure of the woods. This is exactly what our program aims at, the user should see directly how his choices influence the available woods and their ordering. After prioritizing to fire rules as quickly as possible, our system also looks into how to fire as many rules as possible, this is why it looks for the fact that appears the most.
When we call the update() function of our model, the model goes through a couple of steps.

4.5 The update() function of the model

1. The model loops through all the available rules to see if it can fire rules. It checks each rule whether it can fire and if it can, it will fire the rule. When a rule is fired, the model starts to loop again through all the rules, because other rules might be able to fire now due to the change in the knowledge base caused by the rule firing.

2. The system goes through the list of available woods and filters out every wood that was set to unavailable by filtering facts.

3. The system calculates the ordering value of each wood by multiplying the value of each property of the wood by its respective weight. These weights were set by ordering facts that were fired previously. Then the system orders all woods in a descending fashion according to this previously calculated ordering value.

4. Now the system looks for a fact that would maximize the number of rules that the system can fire in the next round, as explained in the subsection above.

5. Finally the system loops through all the questions and compares the facts that are manipulated by each question with the fact that the system wants to know the truth value of. It then counts how often the fact appears in each question and chooses the question in which the fact appears the most. This makes sure that if the knowledge base contains a specific question for that fact, that this question is also chosen as it will contain the fact in the yes and in the no option.

5 User interface, functionality, tools used

The window of our program can be dissected into three different parts. The upper-left part of the window displays the current question that the user should answer, along with a button to reset the program to the start. When there are no more questions that the system needs to ask, then a final message will appear in this window, explaining to the user how the end result came together. The reset button simply resets all the knowledge the system has of the user, therefore it starts asking the first question again.
The lower-left part of the window is rather straightforward: it displays all previous questions and the respective answers of the users.
The last section of the window is the sidebar to the right. In this sidebar all the wood types are shown which are still acceptable according to the answers of the user. The wood types are sorted from the top to the bottom, whereas the top is the most fitting wood type and the bottom the least fitting one. A second important indicator of how well the wood type fits is the color in which the name is printed. The greener the name of the wood type, the better it fits. So the color of the name is quite important to the user, as it stands out even more than the ranking. The color is directly dependent on the ranking score of each wood type compared to the maximum ranking of all wood types. So the color is actually more important than the ranking in the list, as two wood types with the identical color have the same ranking and the more difference there is between colors of two woods, the more unsuitable the redder one of the two is in comparison to the other.

6 Walk-through of a session

As the program is started, the user is greeted by the start of the questioning process, as shown in Figure 2. On the right, the complete list of woods already appear all in green, because no constraint has been yet applied, and so they are currently all fit for the use case.

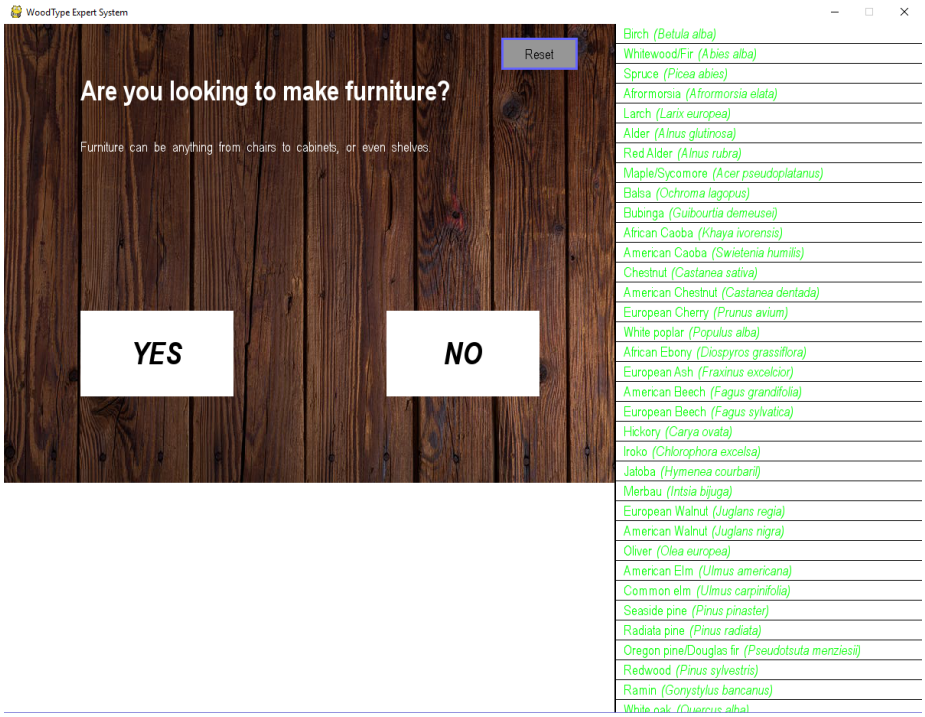

Figure 2: The program at start up.

After answering the first question by clicking on either the "YES" or "NO" button, the question along with the answer appears in the box at the bottom, in order for the user to keep track of what information they already provided. At the same time, the woods in the list start to acquire a shade from green to red, to signify how suitable they are for the task as defined until now. This is shown in Figure 3.

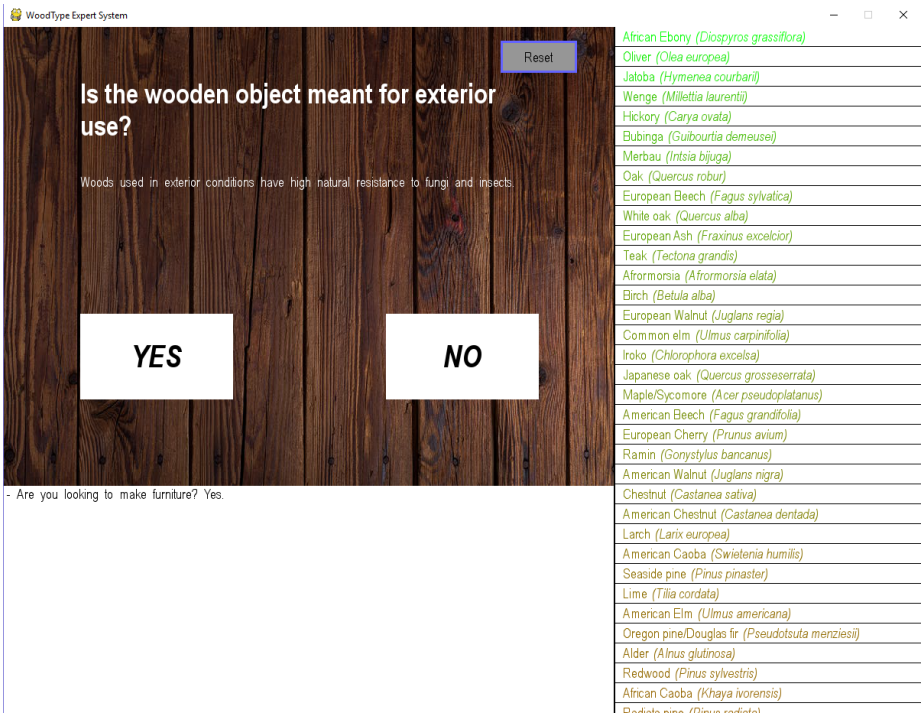

Figure 3: The program after the first question has been answered.

After answering more and more questions, the program has enough elements to filter out many woods that do not comply with the user's requirements. As such, the list of available woods on the right has reduced significantly, and is still showing shades of red and green depending on how fitting the woods are for the task. An example of this is shown in Figure 4.

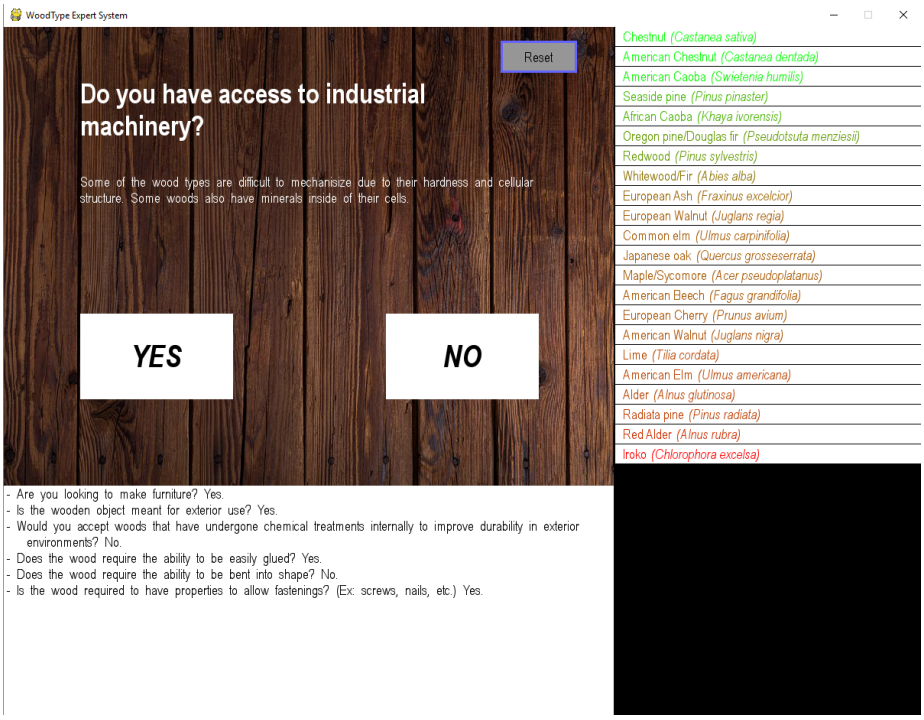

Figure 4: The program after some filters and orderings have been applied.

After having answered all available questions (or at any point during the question answering process), the user can look at the remaining wood types of the list to see which woods have been recommended to him. As seen in Figure 5 the user can hover the mouse over a wood label in the list to get a pop-up with the picture of the appearance of the wood. In the screen the user is also told what conditions the woods have been ordered by based on his answered questions.

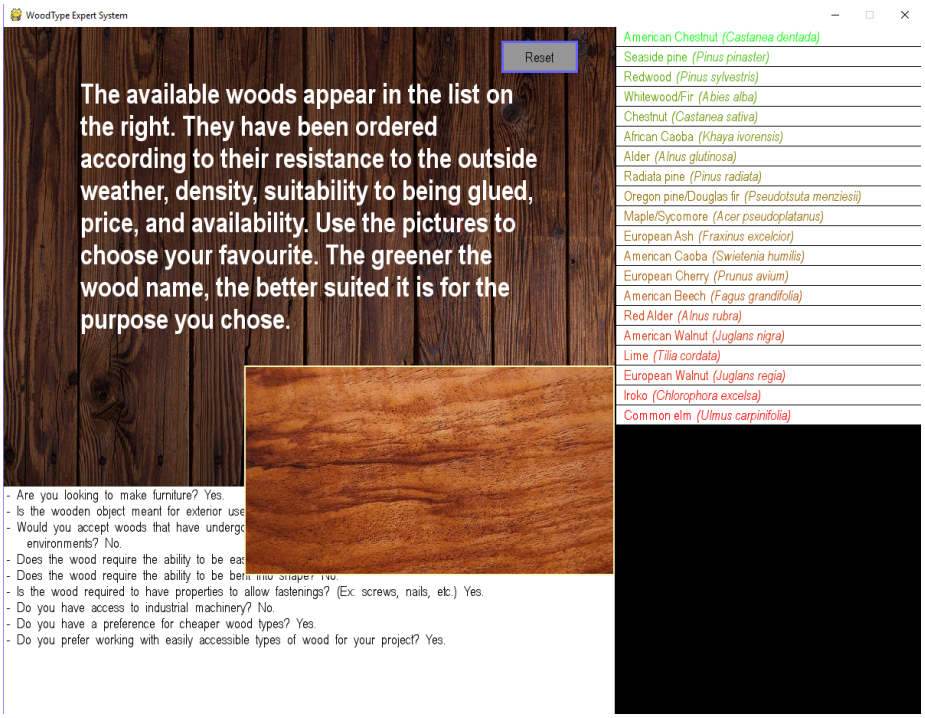

Figure 5: The program at the end of the questioning and inference process, giving the recommendation.

7 Validation of knowledge models

After the 2 initial interviews with the expert, work was done to implement the inference system to reflect the expert knowledge we extracted. Two validation sessions were had with the expert, both of which lasted around 2 hours.
In the first session, we walked him through our inference engine and the rules we had implemented. The expert gave pointers onto which rules we should include and which ones to edit or throw away. The same thing was done with the system's questions. He also gave us a more explicit representation of how his thinking process worked such that we could make more complex rules that better represent him thinking process.
In the second session, he again read through our system's rules to make sure the inference was sound. He also suggested we changed some rules to reorder the way some questions were presented to the user. We furthermore introduced him to the ordering-facts which change the ordering of woods, for which he gave feedback on the weights on the reordering of those individual facts. Over this whole process, we had him play around with the application such that he could give appropriate feedback on the result of the inference system. This

came specially useful for validating that the reordering of the wood types was approximately correct based on the given ordering weights. During his use of the application, he also noticed that certain reordering of woods could make it apparent to a novice user that some woods, although highlighted in deep red, could still be possible to be used for a project, when in practice, no expert would consider it. For this he suggested we changed some ordering oriented fact into a true-false filtering fact. This was the case for data behind Wood Movement and with Hardness of wood. In the case of the user answering "no" to having industrial machinery available, woods such as Teak (one of the hardest to machine woods there are) were still being shown (although in deep red) as our system's ordering does not filter wood types out. Changing this data column to True-False, made it such that no user can fall for this ambiguity by having no hard to machine woods shown at all.

8 Task division

Anton for the most pat worked on the inference engine and the program's model. He was responsible for debugging most of the program's model and inference engine. Nil created the model-controller-view implementation of the program's GUI. He also, due to being a relative of our expert, did a great part on interviewing and improving our program's representation of the expert knowledge. Frank did a lot of work on writing down and implementing our expert's concerns into the application. He also worked on the user interaction side of the application. He worked largely on the report, specially by focusing on making sure we reported every implemented every design choice into it.

9 Reflection

While developing the system we quickly realized that even though the system will be based on simple `if-then` rules, it is still a challenge to implement this properly. Many considerations have to be taken into account. We had to decide what a fact actually is, what values it can have, and how it relates to the questions being asked and to the final conclusion. For implementation purposes, we ended up having three types of facts, two of which had effects on the list of remaining woods. These two types of facts were either ordering-facts (which dealt with numerical values in the data base) or filtering facts (which dealt with True-False values). As mentioned in the validation section above, some numerical data such a Wood Hardness, was changed to True-False to avoid novices thinking non-recommended wood types being shown were a good alternative. This change lost information on wood hardness in the process, which we possibly could have kept by having a fourth type of fact that would both filter and order, such that woods in higher hardness values would be filtered above a certain value and the non-filtered ones could be ordered.
As mentioned by the expert, part decision making that comes with choosing

wood types for a project is going up to the wood storage facility and picking what wood to use by eye based on appearance and color. This was something that a logic-based inference engine could not implement. We thus decided to include this side of the expert's field by adding pictures in the form of pop-ups in our program to give the user a similar experience.
As we also realized further down the road, many of the weights in the ordering-facts were did not combined well as more ordering-facts were implemented. The weights were chosen rather arbitrarily by our expert through a feedback system of trial and error while trying out our application. In the end, the expert seemed quite satisfied with independent orderings of woods based on answered questions, as well as with small combinations of wood ordering-facts, but we came to the conclusion that our procedure of normalizing, multiplying by weights, and adding numerical values in our data base has large room for overfitting and is rather arbitrary for certain larger combinations of ordering-facts.
Furthermore, many of the weights given to such ordering-facts are very volatile due to the data they combine with. Our system was made with data from book of woodworking in Spain. Such values as price and accessibility are very dependant on which part of the world you are in. We hope our system represents overall european values, but that is something we cannot validate, and can vary greatly due to market shifts. The same can be said about (although not to the same extent) about the wood's physical properties' data, as different sources can decide on different values. During the interview with our expert we also learned how fragile such a system is, as it completely depends on the knowledge that we designers interpret from the expert. A mistake of the expert may lead to a disastrous performance, and even more likely are mistakes from our side in interpreting the expert. It took us a while to clearly explain what the goal of our system is, at the start we had a slight misunderstanding, which lead to the expert giving information that is not relevant or helpful in building a knowledge system. From this we learned that it helps a lot when the expert knows exactly what we aim at building and just asking some questions to the expert in an unstructured interview might lead to a knowledge base that does not map on a knowledge-based inference system. Overall this feedback loop is something that, although gets better the more communication there is with the expert, never quite reaches perfection.

10 Link to Github Repository

To run our program, navigate to oxylus/source/dist and then run oxylus.exe. If you have python3 and pygame installed, you can also navigate to oxylus/source and then type in the terminal: "python oxylus.py".

`https://github.com/frank/oxylus/`