

DMN Section 11 Loan Origination Example

By Maarten P.D. Schadd
Senior Product Consultant at Blueriq B.V.

Contents

1	Introduction	3
2	Problem definition	3
3	Designing the process	4
4	Designing the decisions	5
4.1	Decision Requirement Graphs	5
4.2	Decision Logic	5
5	Conclusions	12

1 INTRODUCTION

The decision management community [1] is an initiative that started in 2014 to facilitate the sharing of news and knowledge concerning Decision Management (DM). Next to a product catalog, decision model prototypes and case studies, the decision management community also provides a monthly challenge. Every challenge consists of a problem that should be solved using any business rules and decisions management system or none at all.

As Blueriq is a vendor with an integrated rule engine and decision management capabilities in its BPM suite, we accept this challenge. Blueriq also embraced the decision model [3]. This article describes how Blueriq solves the June 2017 challenge.

2 PROBLEM DEFINITION

This challenge asks for a decision model that is able to make decisions during a loan application. Various decisions are made, including decisions needed for steering the process of the application. As the problem description is long and as it is orderly defined in [2], we do not repeat the complete description here. We invite you to read the full description. Figure 1 shows a complete overview of the decision making of this problem.

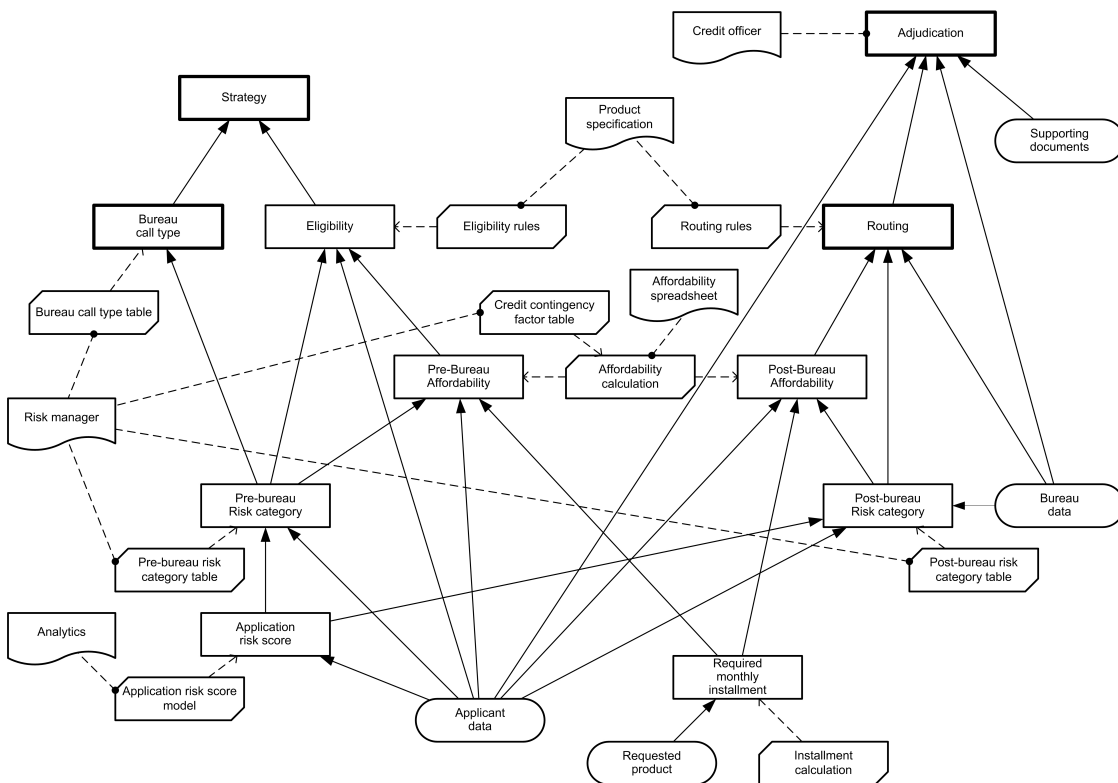


Figure 1: DRD of all automated decision-making

3 DESIGNING THE PROCESS

Blueriq contains a process engine which is able to execute business process. These process usually include manual tasks, performed by knowledge workers or automatic tasks, performed by the system. In the most extreme case a process can be finished completely without needing the knowledge worker, leading to Straight-Through Processing (STP).

In Figure 2 we show the process of Figure 69 in [2] as it could be designed in Blueriq. One big difference is the lacking of decision nodes. Blueriq's backwards-chaining rule engine is invoked each time when a value is needed as long as the value is not known. So if a value is needed in a split node, that is the moment that the rule engine makes the decision. This makes the explicit modeling of a decision moments superfluous.

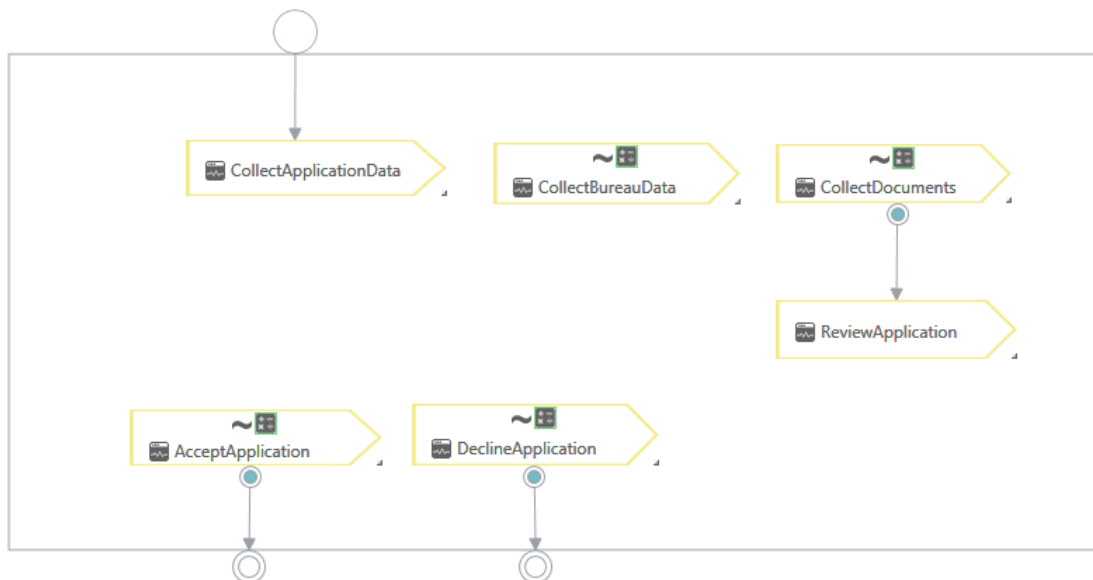


Figure 2: Example business process

We decided to model the process in a manner which resembles the current trend we see in our market. This trend uses much less sequential ordering of tasks and split nodes. It is popular to create the process in a data-driven manner. In the traditional approach you indicate that a task is performed subsequently to another task with a line. We are using ad-hoc tasks which have data driven pre-conditions. As an example, the task ACCEPT APPLICATION has precondition DECISION.ROUTING = "ACCEPT" OR DECISION.ADJUDICATION = "ACCEPT". We indicate when this task is available, and we do not care how the information is obtained. This approach has several advantages and disadvantages. An advantage is that a process can be executed very dynamically. With each new piece of information the process adapts itself and brings itself to the correct state. In the most extreme case in which everything is modelled with preconditions, the process instance could be completely deleted and re-created automatically based on the case data. This mechanism also makes it easy to maintain cases at runtime, as incorrect information can be corrected, and the process adapts itself to the new situation. A disadvantage can be that it may not be clear how the process is executed when looking at the diagram. This is specially true for simple processes. When these become complex, the sheer amount of lines makes them

also difficult to read with the traditional approach.

4 DESIGNING THE DECISIONS

This section describes how we modeled the decisions for the loan application example. All rules, decision tables and logic is executable at runtime, and is created based on our best practices. We follow the ordering as the decisions are presented in [2].

4.1 DECISION REQUIREMENT GRAPHS

In Figure 3 we show the decision requirements graph for the strategy decision. This graph is similar to the one provided in the example. There are some notable differences. (1) The graph is generated from the modelled rules, not drawn manually. This means that it is always up-to-date. (2) It is generated as a tree structure, not as a graph. Certain parts of the tree may be repetitions, as can be seen for the TOTAL RISK decision. For this example we collapsed the details of the righter-most repetition to keep the graph simpler. (3) By default, details regarding the logic or data needed for a decision are hidden. Details can be opened upon request, which results in a view as shown in Figure 4. Here we see that the disposable income is calculated by an unnamed expression, and needs three pieces of raw input data for its calculations. The expression is unnamed as it is set as default expression on the disposable income attribute. The user has the possibility to open any element directly by clicking on nodes in the graph.

Figure 5 shows the decision for the routing of the loan application. This also resembles the example. The tree has one repeating node, POST BUREAU RISK CATEGORY.

Interestingly, the decision for the adjudication does not have a decision requirements graph in Blueriq. There is no logic defined in the example that is needed for this decision. It is made purely by knowledge workers. As the system does not derive any values, no graph is generated automatically.

4.2 DECISION LOGIC

In this section we show all details regarding the decision logic for the load application in Blueriq. Figure 6 shows the decision table for deciding the strategy. In Blueriq, a decision table is read from top to bottom, with the conclusion of the table being the last row. All decisions tables are of type U, Blueriq does not support any other type (C+, C<, C>, C#, C-, P, A, R, F, O, N, ...). One more thing to notice when looking at Figure 6 is the * sign. This represents any value, and is analogue to the - in DMN. The decision alternatives use strings of values, such as "Eligible". Value lists of such values are defined on the corresponding attribute, so that no invalid value can be entered here. Validations are in place to indicate any invalid input, and a combo-box selection can be used for the derived attribute if the user prefers so.

The next table shows the decision for the bureau call type (Figure 7). It speaks for itself and needs no additional explanation.

The eligibility decision table, shown in Figure 8 is an interesting table. The [] represents all values not mentioned already. It uses three independent conditions to make an applicant ineligible. This results in a table with only values on the main diagonal. Another possible manner of modeling this in Blueriq is to set the default value ELIGIBLE on the attribute, and to write three business rules (in the form of IF ... THEN ...) to set the value to INELIGIBLE.

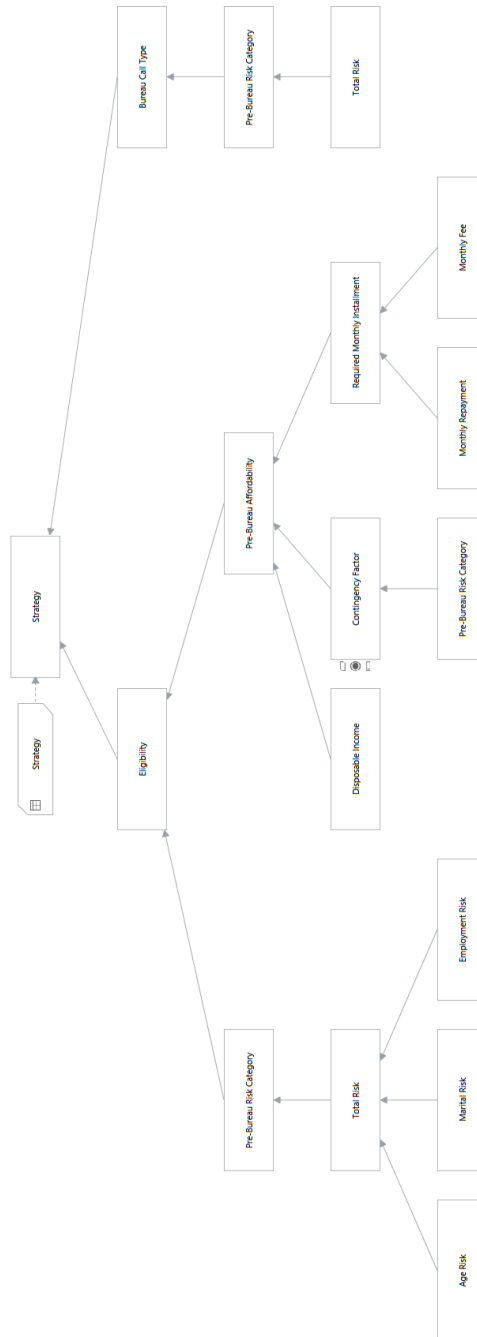


Figure 3: The Strategy Decision



Figure 4: The Disposable Income Decision

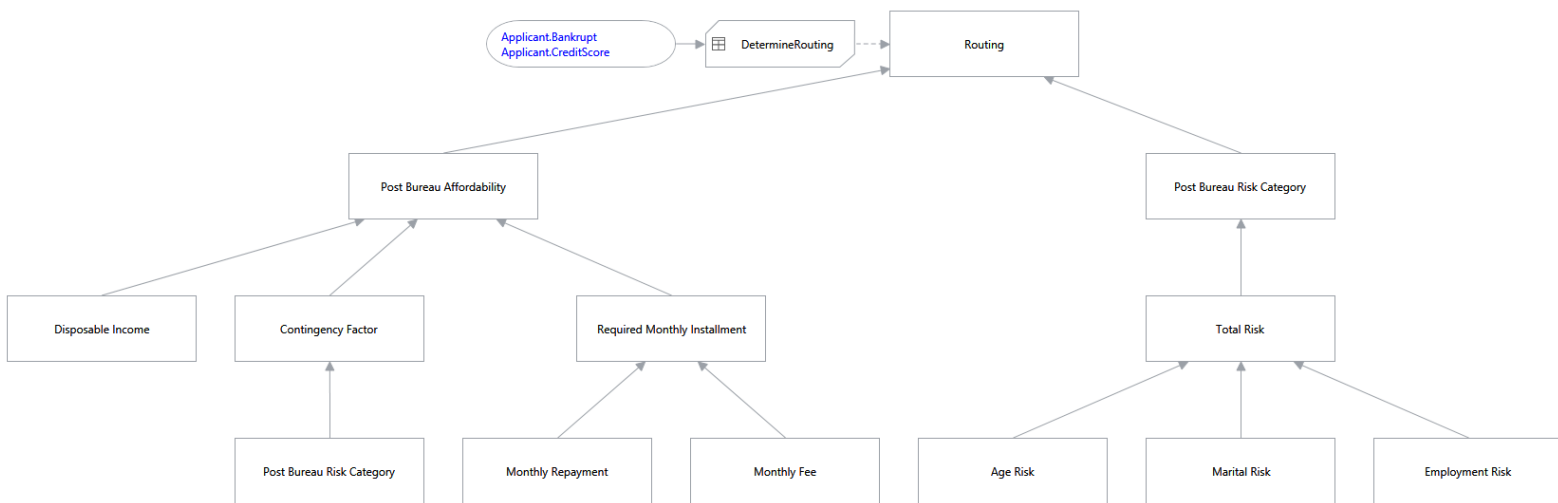


Figure 5: The Routing Decision

Applicant.Eligibility	"Ineligible"	"Eligible"	
Application.BureauCallType	*	"Full" OR "Mini"	"None"
Decision.Strategy	"DECLINE"	"BUREAU"	"THROUGH"

Figure 6: The Strategy Decision Table

Application.PreBureauRiskCategory	"HIGH" OR "MEDIUM"	"LOW"	"VERY LOW" OR "DECLINE"
Application.BureauCallType	"FULL"	"MINI"	"NONE"

Figure 7: The Bureau Call Type Decision Table

This is a matter of taste, and it also has an advantage to have all logic in one place in the decision table. When the decision is dependent on more than five attributes, the table should be split into business rules to avoid a large and mostly empty table.

Application.PreBureauRiskCategory	"DECLINE"	[]	
Application.PreBureauAffordability	*	FALSE	[]
Applicant.Age	*	*	<18 []
Applicant.Eligibility	"INELIGIBLE"	"INELIGIBLE"	"INELIGIBLE" "ELIGIBLE"

Figure 8: The Eligibility Decision Table

The pre bureau risk category table is shown in Figure 9. Please note that borders are sometime inclusive and sometimes exclusive the border value, according to the specifications.

Applicant.ExistingCustomer	TRUE				FALSE			
Application.RiskScore	<100	>=100 AND <120	>=120 AND <=130	>130	<80	>=80 AND <90	>=90 AND <=110	>110
Application.PreBureauRiskCategory	"HIGH"	"MEDIUM"	"LOW"	"VERY LOW"	"DECLINE"	"HIGH"	"MEDIUM"	"LOW"



Figure 9: The Pre Bureau Risk Category Decision Table


The next decision concerns the risk model score of a customer that applies for a loan. This decision table is of type C+ of the DMN standard. This indicates that all outcomes are tried, and any match is accumulated to a grant total value. This type of decision table is frequently used for calculating a score which is based on several independent factors that each change the value independent of the other factors. When designing such a decision table, one has to be careful to have exclusive columns as otherwise the understandability of the table is low.


In Blueriq, decision tables of type C+ are not supported. All tables are of type U. There are however advantages of modeling this decision with only this type of tables. Advantages are (1) nicely separated logic for each decision category, (2) smaller decision tables that do not have many empty cells, (3) easier development and better maintainability as the calculation of the score is broken down into smaller chunks. Disadvantages are (1) additional helper attributes in the domain and (2) the logic for one business decision is split up into different tables.

The total application risk score is calculated as default expression on the attribute as shown in Figure 10. The separate calculations for age, marital status and employment status are shown in Figures 11, 12 and 13, respectively.


The decision for the routing is shown in Figure 14. Just as in [2], the table is largely empty. This is due to that four factors decide the outcome in a boolean manner. This table is of type P, and we had to rearrange the table slightly to work but it is still readable and maintainable.

DEFINITION  



Entity  Application

Name RiskScore  Askable

Functional name Total Risk

Type 12 Integer  MultiValued

☐ Acts as Reference

Value list  

DEFAULT VALUE

Type Expression

Expression

```
Application.AgeRiskScore + Application.MaritalRiskScore + Application.EmploymentRiskScore
```

Figure 10: The total application risk score



Applicant.Age	>=18 AND <=21	>=22 AND <=25	>=26 AND <=35	>=36 AND <=49	>=50
Application.AgeRiskScore   	32	35	40	43	48

Figure 11: The age risk score




Applicant.MaritalStatus	"S"	"M"
Application.MaritalRiskScore   	25	45

Figure 12: The marital risk score




Applicant.EmploymentStatus	"UNEMPLOYED"	"STUDENT"	"EMPLOYED"	"SELF-EMPLOYED"
Application.EmploymentRiskScore   	15	18	45	36

Figure 13: The employment risk score

Application.PostBureauAffordability	FALSE			
Applicant.Bankrupt	*	TRUE		
Application.PostBureauRiskCategory	*	*	"HIGH"	
Applicant.CreditScore	*	*	*	<580
Decision.Routing		DECLINE	DECLINE	REFER
		DECLINE	REFER	ACCEPT

Figure 14: The routing rules decision logic

Figure 15 shows the post bureau risk category decision table. We decided to split the table to improve the readability in this article. It actually is one large decision table but it is too wide to include it completely here. It is possible to model it exactly as shown in Figure 15 in Blueriq. You then have two tables which each have a single option in the first row. This means that both tables are not complete. The rule engine tries both tables to derive a result, and as long as there is no contradiction, this works as expected. Splitting up decision tables in this manner can greatly improve the readability of large tables at the cost of not seeing all the logic in one single table. The decision requirements graph shows then both tables for this decision.

Applicant.ExistingCustomer	FALSE						
Application.RiskScore	<120		>=120 AND <=130			>130	
Applicant.CreditScore	<590	>=590 AND <=610	>610	<600	>=600 AND <=625	>625	*
Application.PostBureauRiskCategory	HIGH	MEDIUM	LOW	HIGH	MEDIUM	LOW	VERY LOW

Applicant.ExistingCustomer	TRUE					
Application.RiskScore	<=100			>100		
Applicant.CreditScore	<580	>=580 AND <=600	>600	<590	>=590 AND <=615	>615
Application.PostBureauRiskCategory	HIGH	MEDIUM	LOW	HIGH	MEDIUM	LOW

Figure 15: The post-bureau risk category table decision logic

The next piece of interesting logic concerns the pre- and post-bureau affordability. This is challenging as logic uses a generic risk category which could be pre- or post-bureau risk category, depending on what type of decision we are trying to take. This resembles a function call with one parameter. The concept of a function is not something that is currently available in Blueriq. We are currently looking into it as a mechanism to decouple the business model by using function calls. As we are not there yet, we need to make a small workaround to make this scenario work. The business rules for the pre- and post-bureau affordability are shown in Figures 16 and 17, respectively. Both attributes have a default value of FALSE, which is overwritten by the business rules if the IF clause evaluates to TRUE.

As you can see, we have created two attributes for the pre- and post-credit contingency factors. Depending on which variant is calculated, the appropriate attribute is included in the logic. The logic for these attributes is shown in Figures 18 and 19. These tables are rather similar, just with

IF	<code>Applicant.DisposableIncome * Application.PreCreditContingencyFactor > Application.RequiredMonthlyInstallment</code>	
THEN	Application	PreBureauAffordability
IS	TRUE	

Figure 16: The pre bureau affordability decision logic

IF	<code>Applicant.DisposableIncome * Application.PostCreditContingencyFactor > Application.RequiredMonthlyInstallment</code>	
THEN	Application	PostBureauAffordability
IS	TRUE	

Figure 17: The post bureau affordability decision logic

different in- and outputs. I agree with the argumentation that this duplication of logic is not maintainable, as any change has to be performed multiple times and a mistake is quickly made. That is why we want to change this in the near future.

Application.PreBureauRiskCategory	"HIGH" OR "DECLINE"	"MEDIUM"	"LOW" OR "VERY LOW"
Application.PreCreditContingencyFactor	0.6	0.7	0.8

Figure 18: The pre credit contingency factor decision logic

Application.PostBureauRiskCategory	"HIGH" OR "DECLINE"	"MEDIUM"	"LOW" OR "VERY LOW"
Application.PostCreditContingencyFactor	0.6	0.7	0.8



Figure 19: The post credit contingency factor decision logic


Last, but not least, the installment calculation is shown here. It falls apart in several steps. The installment attribute is shown in Figure 20, which simply adds up the two components with a default expression.

The attribute representing the monthly repayment (not shown) has this default expression: PMT. This is a reusable expression defined as

`(Loan.Amount*Loan.Rate/12)/(1-(1+Loan.Rate/12)**-Loan.Term)`, and can be used anywhere.


While this resembles a function call, it has no parameters, so we can not use this mechanism for the affordability calculation (Figures 16, 17, 18 and 19). As the PMT calculation is not used anywhere else, it can also be placed as a default expression directly on the attribute. As this



DEFINITION  

Entity  Application

Name RequiredMonthlyInstallment ☒ Askable

Functional name Required Monthly Installment

Type  Currency ☐ MultiValued

Value list  

DEFAULT VALUE

Type Expression

Expression `Loan.MonthlyRepayment + Loan.MonthlyFee`

Figure 20: The monthly installment decision logic

calculation is a known term for the business, we decided to model this in the shown manner. The monthly fee is calculated in the small decision table shown in Figure 21.

Product.Type	"STANDARD LOAN"	"SPECIAL LOAN"
Loan.MonthlyFee   	20	25

Figure 21: The monthly fee decision logic

5 CONCLUSIONS

Blueriq is able to create an executable decision model from the provided loan application specifications. A decision requirements graph can be constructed during design and runtime [7] that shows all dependencies of the decision. While not all concepts of DMN are available in Blueriq, an executable and maintainable model can still be constructed. For more information on the capabilities of Blueriq and examples, we refer to these articles: [3, 4, 5, 6, 7].

CONTACT US

If you have any questions about this article or if you would like to start a discussion, do not hesitate to contact us.

Email the author : m.schadd@everest.nl
Email Blueriq : info@blueriq.com
Call Blueriq : +31 (0)73 6450467
Website Blueriq : <http://www.blueriq.com>

Blueriq BV
Veemarktkade 8
5222 AE s-Hertogenbosch
The Netherlands

ABOUT BLUERIQ

Blueriq is a rule-driven software platform designed to deliver dynamic business solutions for organizations with knowledge-intensive processes. It empowers organizations in fast changing environments to quickly and cost-effectively respond to changing business conditions and regulations. Blueriq provides solutions for Decision Management, Dynamic Case Management and intelligent User Experience Management across multiple channels. Solutions based on Blueriq are modeled, not programmed, giving you the opportunity to respond more quickly to your customers needs and improving your business outcomes. With Blueriq, you make your own rules!

©2017 Blueriq B.V. All rights reserved.

REFERENCES

- [1] Decision Management Community. <https://dmcommunity.wordpress.com/home/>, 2014.
- [2] Object Management Group. Decision Model and Notation (DMN), version 1.1. formal/2016-06-01, 2016.
- [3] M. P. D. Schadd. Blueriq embraces the decision model. Technical report, Blueriq B.V., 2013. <http://schadd.com/Papers/2013dmn.pdf>.
- [4] M.P.D. Schadd. Case study: Vehicle insurance user product derby. Technical report, Blueriq B.V., 's-Hertogenbosch, The Netherlands, 2015. <http://schadd.com/Papers/2015DMCVehicleInsurance.pdf>.
- [5] M.P.D. Schadd. Collections of cars. Technical report, Blueriq B.V., 's-Hertogenbosch, The Netherlands, 2015. <http://schadd.com/Papers/2015DMCCollectionOfCars.pdf>.
- [6] M.P.D. Schadd. Decision table for vacation days calculation. Technical report, Blueriq B.V., 's-Hertogenbosch, The Netherlands, 2016. <http://schadd.com/Papers/2016DMCVacationDays.pdf>.
- [7] M.P.D. Schadd. Port clearance rules. Technical report, Blueriq B.V., 's-Hertogenbosch, The Netherlands, 2016. <http://schadd.com/Papers/2016DMCPortClearance.pdf>.