Decision Exploration Lab: A Visual Analytics Solution for Decision Management

Bertjan Broeksema, Thomas Baudel, Alex Telea, and Paolo Crisafulli

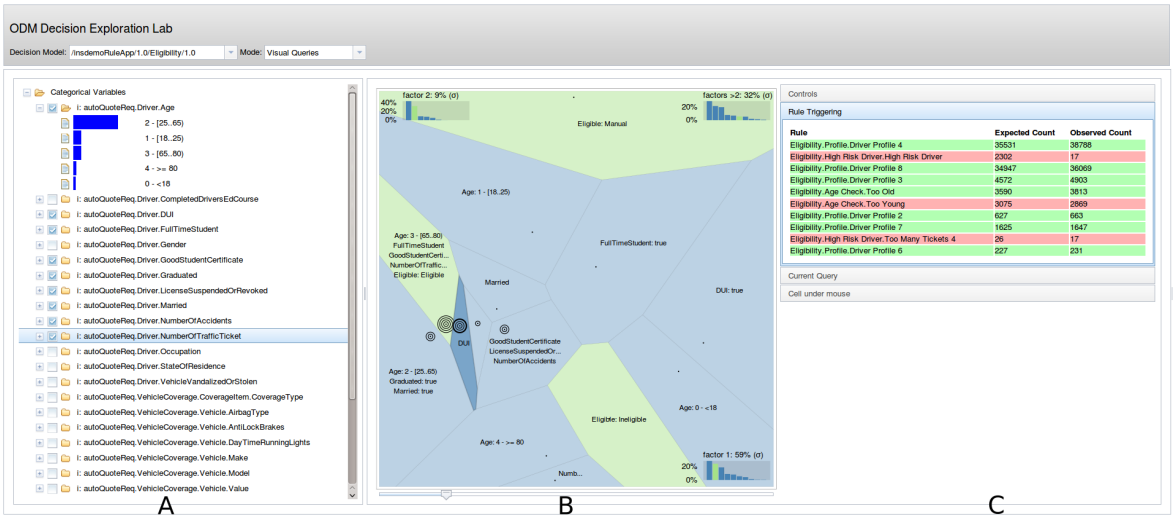


Fig. 1. Decision Exploration Lab in visual querying mode, with the *DataSpace tree* (A), the *Decision Map* (B) and the *Rule Triggering view* (C).

Abstract—We present a visual analytics solution designed to address prevalent issues in the area of Operational Decision Management (ODM). In ODM, which has its roots in Artificial Intelligence (Expert Systems) and Management Science, it is increasingly important to align business decisions with business goals. In our work, we consider decision models (executable models of the business domain) as ontologies that describe the business domain, and production rules that describe the business logic of decisions to be made over this ontology. Executing a decision model produces an accumulation of decisions made over time for individual cases. We are interested, first, to get insight in the decision logic and the accumulated facts by themselves. Secondly and more importantly, we want to see how the accumulated facts reveal potential divergences between the reality as captured by the decision model, and the reality as captured by the executed decisions. We illustrate the motivation, added value for visual analytics, and our proposed solution and tooling through a business case from the car insurance industry.

Index Terms—Decision support systems, model validation and analysis, multivariate Statistics, program analysis

1 Introduction

Complexity of legislation and changing environments, markets, and business policies render the making of informed business decisions increasingly hard. Many operational decisions are made at the bottom of organizations by people who do not have the full knowledge or access to information needed for optimal decision-making. To alleviate this, expert systems have been integrated in enterprises to help automating complex decisions [26, 34]. A key feature of expert systems is the separation of knowledge *representation* and knowledge *execution*. However effective, expert systems are typically used to solve one particular problem, rather than a wide range of problems.

Enterprise Decision Management (EDM) takes a more generic approach to decision automation by means of Business Rule Management Systems (BRMSs) and Decision Management Systems (DMSs) [45]. BRMSs are tailored to model and execute a wide range of business decisions. Atop of knowledge modeling and basic execution management, DMS add monitoring and performance analysis of BRM automated decision execution.

Decision models (DMs) express the workflow and processes of an enterprise. They are used for both high-complexity decisions and everyday high-volume operational decisions. DMs enable non-technical business users to model the business domain and business logic instead of providing requirements to IT departments that have to implement such models. The result is a body of *explicit knowledge*, *i.e.*, an ontology modeling the business domain and a rule-set modeling the business logic respectively. The combination of explicit knowledge, automated decisions based on this knowledge, and the sheer amount of decisions being made by DMSs, bring many challenges to business analysts that use such systems.

Information visualization (infovis) and visual analytics (VA) study how graphical representations and interaction, as external representations, support the internal processes of reasoning, understanding, and sense making [44, 21]. EDM is an interesting case for infovis and VA since it makes part of the internal representation and reasoning mech-

- *Bertjan Broeksema is with IBM France Center for Advanced Studies, Institute Johann Bernoulli, Univ. of Groningen, The Netherlands and INRIA, University of Bordeaux, France. E-mail: bertjan.broeksema@fr.ibm.com.*
- *Thomas Baudel is with IBM France Center for Advanced Studies. E-mail: baudelth@fr.ibm.com.*
- *Alexandru C. Telea is with Institute Johann Bernoulli, Univ. of Groningen, The Netherlands. E-mail: a.c.telea@rug.nl*
- *Paolo Crisafulli is with IBM France. E-mail: pcrisafulli@fr.ibm.com*

Manuscript received 31 March 2013; accepted 1 August 2013; posted online 13 October 2013; mailed on 4 October 2013.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

1077-2626/13/$31.00 © 2013 IEEE Published by the IEEE Computer Society

anisms external, and hence available for explicit analysis, and it also exposes facts on the performed decisions.

In this paper, we present the Decision Exploration Lab (DEL), a set of infovis and VA tools that help business users to explore the knowledge and execution facts gathered from a BRMS. Following the task-oriented framework of Maletic *et al.* [36], we can describe DEL as follows: The targeted *tasks* are to get an overview of the decision logic and domain ontology and to find correlations and potential discrepancies between these facts and actual executions of the decision logic; the *audience* is formed by EDM business analysts who understand the modeled domain well, but are not IT nor BRMS experts; the analysis *target* is the domain modeling (ontology and decision rules) and the facts gathered from rule execution; the visual *representation* consists of a set of linked views (scatter plots, tree views, and bar charts); finally, the presentation *medium* is the standard 2D PC screen.

The structure of this paper is as follows. In Sec. 2 we overview decision management and automation, and the challenges that advocate a visual analytics approach. In Sec. 3 we identify the core tasks that we want to support and introduce the Decision Exploration Lab (DEL), a visual analytics tool that helps performing these tasks. In Sec. 4 we show how DEL supports the identified tasks using a scenario from the car insurance industry. Sec. 5 discusses the implementation of DEL. Section 6 concludes the paper with a discussion of DEL both from a research and a product standpoint.

2 What is Decision Automation?

Decisions are an increasingly important asset in modern business. In highly competitive complex and rapidly changing markets, business decisions should be traceable and adaptable [35, 30]. Decision automation is one way to reach this, by encoding business knowledge in an executable decision model [39]. Creating a decision model is the task of a business analyst, or more generally a knowledge engineer.

Let us consider a scenario from the car insurance industry: handling car insurance requests. Two decision models are used here: risk management (DM_1) and pricing (DM_2). Figure 2 shows this scenario using the well-known BPMN business process notation [40]. Both models have as input an insurance request, which contains data about the driver, the car, and the requested insurance. The risk management model DM_1 first determines if a driver is eligible for an insurance. Its output is one of the following: *eligible* (the system could automatically find that an insurance can be given); *ineligible* (the system found reasons to deny the request); or *manual processing* (the system could not take a decision, so a human operator should intervene). Once DM_1 has determined eligibility, the pricing model DM_2 is triggered to compute the insurance base quote, a list of discounts and surcharges, and the final insurance quote. We next use this scenario as a running example for our paper and discuss it in more detail in Sec. 4.

Each decision model (DM_1 and DM_2) has two parts. First, an *ontology* describes the objects subject to decision, their attributes, and the decision outcomes. In our example, such ontologies have terms such as insurance (with quote, surcharges, and discounts attributes), requesters (with person detail attributes, *e.g.*, age, driving history, and salary), and cars (with technical vehicle details as attributes). Ontologies can have a hierarchical structure modeling the natural grouping of concepts or types. Secondly, a decision model has a set of *production rules* that describe how the decision is taken on an ontology instance. Rules have a precondition (IF statement) and an action that is executed when the precondition is met (THEN statement). For example:

```
IF
   the vehicle has antilock brakes
THEN
   add a 5% discount to the quote
```

A decision model encodes both (1) well-known (unambiguous) external factors and (2) expectations (or assumptions) about the decision's input domain. In case (1), we have rules that enforce regulations or physical constraints, *i.e.*, aspects of reality that must be taken into account. For example, a rule states that no requester without a driving license can be offered a car insurance, since the law requires a license to drive in the first place. As an example of case (2), consider the rule:

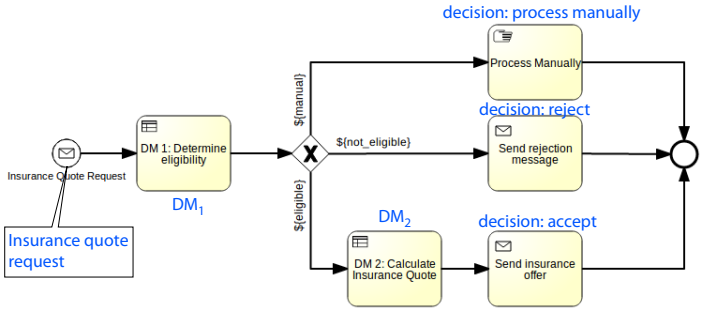


Fig. 2. A car insurance request, containing data about the driver, vehicle, and requested insurance is processed by two decision models (DM_1, DM_2). It results in a pricing offer, rejection, or manual processing.

```
IF
   the driver has been caught for driving under influence
THEN
   set the status of the insurance request to ineligible
```

This rule encodes the assumption that the number of insurance requesters caught for driving under influence (DUI) is low enough to ensure that this rule will not neglect too many requesters. Another way to deal with DUI cases is accepting them but assuming a higher accident risk and thus giving them a higher quote. Assumptions are made in both cases but dealing with these assumptions follows different paths. A challenging problem is to find how well such assumptions match the actual decisions that have been made over time.

Decision models are only *partial* models of the reality. When creating them, business analysts use their experience to decide which knowledge should be explicitly captured (in ontology objects and associated rules) and which knowledge can be left out. For complex models consisting of hundreds or thousands of rules it is very hard or even impossible to know if all such assumptions are correct and what their interplay is.

The problem we address next generalizes the above observations, and can be summarized as follows: How can we explore and understand how reality as captured by decision models *diverges* from reality encoded by actual decisions taken over time?

2.1 Decision Management Systems

The value proposition of decision management systems is that they separate the business *logic* from the enterprise *applications* implementing a business process. Such applications send an instance of the ontology that serves as input for a decision model to the decision service and retrieve the decision for further processing. We next give a high-level overview of BRMS's, based on IBM's Operational Decision Manager (ODM) [25]. The presented concepts apply also to other systems such as Drools [27] and FICO Blaze Advizor [14].

The main components of a BRMS relevant to our context are decision model repositories, execution engines, and operational data stores. Decision models are stored in a BRMS repository. A front-end accesses the repository to provide rule authoring, access management, version handling, and model deployment. Once a model is deployed, it provides a decision *service* to all enterprise applications using it.

A decision model has input and output parameters, both types being described by the decision model ontology attributes. Input parameters model the information based on which decisions are taken, *e.g.*, the details on the person requesting an insurance and the vehicle parameters. Output parameters model the actual decision, *e.g.*, the decision to grant an insurance and the insurance details. Business logic, or the way to arrive to an output from a given input, is captured by *production rules* which have unique names and are structured in packages. These rules can be expressed in several forms, *e.g.* controlled natural language [11] and decision tables.

A rule-execution engine performs the decision logic for the given input ontology instance. Such instances are also known under the

name of individuals in artificial intelligence and observations in statistics. Given an instance, the engine finds which rules need to be triggered, and in which order, to reach the final decision. For this, inference algorithms are used based on variants of RETE or TREAT [16, 38]. The decision model's output parameters, *i.e.* the actual decision, are set to the decided values by the action part of the triggered rules. All inputs, decisions (outputs), and rule execution traces are stored in the operational data store of a BRMS.

When analyzing automated decisions, all above data sources need to be considered. Hence, such analyses involve three data spaces:

- **DS_1 - Decision model:** The ontology and production rules operating on this ontology that describe how decisions are made.
- **DS_2 - Decision instances:** Instances of the ontology for which a decision has been made according to the decision model.
- **DS_3 - Decision execution traces:** A trace of triggered production rules for each decision.

These data spaces have some analogy to well-known data sets in software engineering [9]: Type declarations, or UML models, in a program are analogous to our decision model ontologies. The imperative program source code is analogous to our production rules. Variable values in a program execution trace are analogous to decision instances. Finally, sequences of executed statements from a program execution trace mirror our decision execution traces. At a higher level, our goal of understanding how a set of decisions has been reached resembles program comprehension goals in reverse engineering and dynamic analysis [8].

2.2 Analytics requirements for Decision Management

Each of the three data spaces $DS_1..DS_3$ adds to the challenge of understanding automated decisions in its own way. When building a decision model (DS_1), analysts use their context knowledge to determine which information is important to make an informed decision. Only this information will be, thus, encoded in production rules. The technical details of rule writing are here less relevant. What is crucial, however, for analysts is to understand how their modeled decisions impact the business and if their execution aligns with business goals.

In this context, several questions are of interest to business analysts:

- What is the structure of input data in terms of related concepts?
- How do input concepts relate to the decision?
- Which parts of the decision logic are relevant for a certain selection of decisions?

For these questions, we further identify two types of challenges.

Scale: Typical BRMS ontologies are complex, consisting of many concepts, which in turn are a mix of categorical, numerical and temporal variables. Decision automation is usually applied in areas that have a high throughput. This leads to a large collection of facts with a high complexity that has a strong relation with the business logic. For instance, the ontology for the decision models from the car insurance scenario in this paper consists of 22 objects with a total of 94 attributes. The associated rule-set has 166 rules. A typical run of this rule-set over an input-set of requesters generates over 100K decisions.

Dataset sizes in industrial applications of BRMS's are much larger. For instance, IBM consultants worked on a decision model for a loan industry application, consisting of 100 concepts with an average of 20 attributes per concept; this model's business logic has 2600 production rules. The throughput for this decision service is of hundreds of executions per second. For a credit-card use-case, the decision model has 1300 concepts. This model consists of multiple rule-sets each having between 10K and 20K rules. The throughput for this decision service is of 2400 decisions per second on average, with peak days of 55 million decisions.

Performance measuring: In IBM ODM [25], the performance of a decision is measured by so-called key performance indicators (KPIs). Monitoring KPIs over time yields so-called business performance trends. A KPI is a function of an ontology's attributes, computed over ontology instances, and resulting in a single value. KPIs can be defined for input parameters, *e.g.*, the male-female ratio of car insurance requesters, and output parameters, *e.g.*, the fraction of eligible car insurance requests from the total request set. Input KPIs cannot be influenced by changing the decision model. However, output KPIs are defined by the interplay between the decision model and decision instances.

Output KPIs are similar to metrics used to assess the quality of a software product or process, *e.g.* lines of codes, number of bugs found, or number of passed tests [12]. Many tools exist for visual analysis of such metrics [33]. These could be directly used in our context. However, a major issue exists with output KPIs, both in the software industry and in our context: While metrics show process aggregated performance, they do not explain *why* that performance has been reached. For instance, in the software industry, we can monitor the bug discovery rate, but this metric will not tell why more (or less) bugs have been introduced. The same is true for output KPIs in our context: These show increase or decrease of business performance, but give few or no clues on why this variation is happening. Also, typical KPIs say little about the relation between current *vs* optimal performance, which is an important business driver [35, 30].

Core tasks: The above observations lead to the following high-level tasks that we want to address with our approach:

- **T1:** Create selections of decisions for performance monitoring, validation, auditing and testing purposes.
- **T2:** Understand correlations between input data and decisions.
- **T3:** Verify expected and discover unexpected operation of a decision system.
- **T4:** Find KPI-related opportunities for improving existing decisions.

2.3 Visual analytics solutions

Knowledge modeling, representation, reasoning over knowledge, and consistency checking are closely related topics to our problem domain. A related topic is machine learning, where models are built from observations. In our case, we have an existing model expressed as IF-THEN production rules and want to use observations to discover how *suitable* this model is for a given observation set. For more details, we refer to Russell *et al.* [41].

Software visualization: Software visualization aims to depict the structure, behavior and evolution of software [9]. As outlined in Sec. 2.1, several similarities exist between the life-cycle of software systems and decision models, and also between the involved data structures. However, the questions for DMSs (Sec. 2.2) are quite different from typical software comprehension goals. These are due to the different execution model of rule-sets compared to software systems. The rule execution order is a side-effect of the execution engine and thus has no particular meaning to a business analyst. In contrast, imperative software executes statements in the order they appear in the source code. More importantly, our core question is different: We want to see how the realities captured by the decision model and the executed decisions diverge. This question is of a higher level than what typical software comprehension tools, such as dependency graphs and executions charts, directly address.

Data modeling: Most business ontology attributes are of categorical type. To analyze such data, two main approaches exist [13]: In the area of categorical visualization (CatVis) methods, Friendly *et al.* present several visualization techniques, with a focus on mosaic plots [17, 18, 19]. Other CatVis techniques include treemaps [28], CatTrees [31], parallel sets [32], and the contingency wheel [3]. Although these

methods are specifically designed for categorical data, they are also more focused on, and effective for, *frequency-related* tasks. Quantization methods, on the other hand, model categories by numerical values and are effective for *similarity analysis* tasks. Such methods reduce a large number of dimensions to a number that is suitable for display, *i.e.*, two or three. An overview of such methods is given by Fodor [15]. In our work, we focus on a subclass of dimensionality reduction methods known as Multiple Correspondence Analysis (MCA). This technique has been known under other names [46] and is particularly suitable for categorical data, as opposed to the more general multidimensional scaling (MDS) techniques which address numerical data. MCA results are usually visualized with 2D or 3D scatterplots. CA maps [22] map each category to a plot point. (CA) biplots [20, 22] extend CA maps by mapping both categories and observations to plot points. For the exploration of ontology data, visualization methods using node-link displays of the ontology graph can be used [29].

Concluding, several visualization techniques exist for multivariate data and ontologies. However, none of these is a direct and full solution for all our challenges. What we need is not just to show similarity relations between data points, but also cause-effect and correlation-effect relations – for instance, which rules cause a certain decision and how a correlation of values influences the rule triggering.

2.4 Decision support VA tools

Savikhin *et al.* [43] present a visual tool that helps users make informed auction bids to overcome the winner's and loser's curses. A graph shows the theoretical gain/loss predictions for a given bid, informing the user about the most appropriate bid. PortfolioCompare is a visual analytics tool that supports financial planning decisions for financial investment portfolios [42]. In this tool, users can add multiple financial portfolios and analyze risk and return aspects of each portfolio. AlHajj *et al.* present a Tableau-based visualization for decision-making support in treating injuries [2]. All above tools focus on improving *human* decisions. In contrast, we want to give a human insight in, and ways to improve, *automated* decisions. Afzal *et al.* [1] describe a system for interactive what-if analysis of the impact of decisions to fight epidemic spread. A history tree view shows the lost or saved lives over time as a function of taken decisions. This helps analyzing the relation between a set of decisions and the outcome, where outcome is considered to be one variable. Migut and Worring present an interactive risk assessment framework using 2D scatterplots, Voronoi diagrams, and treemaps to visually explore the results of an automatic classifier, highlighting cost boundaries for different trade-offs [37]. We also use Voronoi diagrams and 2D scatterplots as visual tools, but our focus is different: Explore the decisions made by a user-programmed rule-based system (as opposed to a trained classifier), and see how rules and decisions interact. We are interested to see the aggregated effects of the same decision made potentially millions of times. Thus, we do not focus on a sequence of different decisions but on one particular decision and consider that each decision has its own output variables. This is opposed to [1] where all decisions influence the same output variable. Decision rules are also visualized by Wlodyka *et al.* using node-link diagrams and 3D matrices [48]. However, their focus is only to show the rule-sets, and not their effects on actual decisions. Wang *et al.* present a methodology and workflow for designing VA systems for decision support in organizations [47]. However, this general framework does not focus on the concrete challenges for BRMS rule-based systems.

3 The Decision Exploration Lab

In the following, we present the Decision Exploration Lab (DEL), a toolset for combined textual and visual analysis of DMS datasets. The design of DEL aims to provide direct support for the core tasks related to DMSs as outlined in Sec. 2.2. We start with a general introduction of the components of DEL (Secs. 3.1 and 3.2). Next, we discuss several tool refinements based on its actual utilization (Sec. 3.3). In Sec. 4 we show how DEL supports our core tasks, based on two actual user stories involving our car insurance scenario.

3.1 Querying and filtering

Many decision analysis scenarios start with narrowing down the search space. When troubleshooting a particular decision, business analysts need to find the decision for a person with a given name and ID. A more challenging use-case involves analysts wanting to focus on a particular segment, *e.g.* persons with a certain education level or above-average-risk credit card transactions. We support such use-cases by a a query and filter mechanism. Since DMS analysts are used to express production rules in controlled natural language [11], we extended this approach to allow querying decisions. Concretely, users can query decisions based on both the checked preconditions and actions taken by production rules, in natural language. Two examples follow:

```
find decisions where the vehicle has antilock brakes;
find decisions where the quote has at least a 5% discount;
```

Recall that a decision has three parts: an ontology instance representing the input; an execution trace; and an ontology instance representing the decision. When a query is entered (Fig. 3A), a table listing decision details is updated (Fig. 3B). The user can add or remove ontology attributes to this table to show only the information required for the task at hand. Selecing a decision from the table opens a separate view showing details of the selected decision. For example, Fig. 3C shows the detail view for rules triggered for a student who was not eligible for a car insurance.

This approach covers simple queries on the decision warehouse when users know what to ask. Use-cases include auditing and troubleshooting for which details of single decisions must be known and the decision itself can be reasonably easy identified by known criteria, and creating selections of decisions for monitoring or validation (**T1**). We also use this function as a preliminary filter for our visual exploration mode (see Sec. 3.2 next). Specifically, users employ queries to reduce the set of decision instances in a way that is most suitable for the kind of questions he wants to answer next.

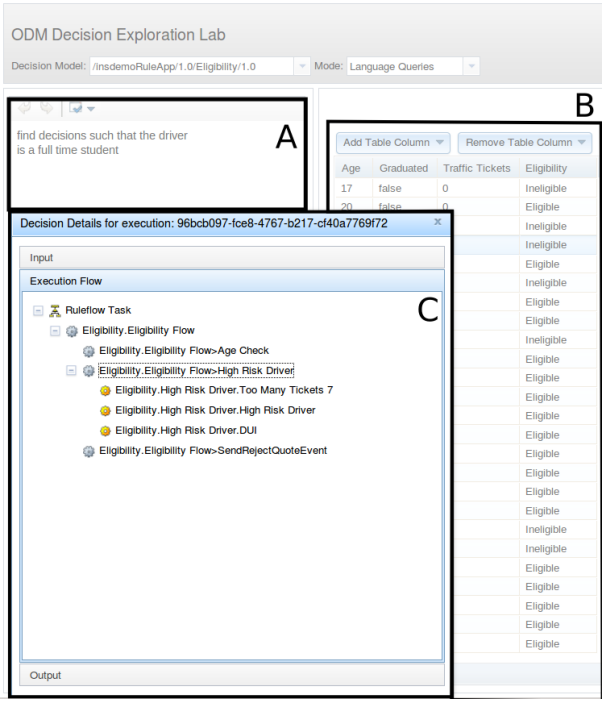


Fig. 3. Textual query view (A), with query result table (B), and detail view (C) showing triggered rules for a selected decision.

3.2 Visual Decision Exploration

The visual decision exploration mode of DEL (Fig. 1) helps finding two types of relations:

1. Relations between ontology concepts themselves.

2. Relations between ontology concepts and the decision logic.

This view has three main components: The *DataSpace tree* shows all ontology variables. The *Decision Map* helps understanding correlations between concepts. The *Rule Triggering view* shows how production rules map to actual executions. The *DataSpace tree* and *Decision Map* build upon the techniques presented in Broeksema *et al.* [6]. These techniques have been initially introduced in the general context of analyzing multivariate categorical data. We adapted and extended these techniques to our more specific context of analyzing ontologies, rule sets, and the execution thereof, as follows.

3.2.1 DataSpace tree

To support analysis on ontology instances from the decision model, we use a tree view, as follows (see Fig. 1A). Each ontology attribute of basic data type (*e.g.* boolean, string, numeric) becomes a tree node. Parent nodes are created following the object hierarchy. For example, the input parameter *InsuranceRequest* has a property of type *Person*, which in turn has a numeric attribute *age*. This yields a branch *input.InsuranceRequest.Person.age* in our tree.

We separate variables in three subtrees: categorical, numerical and temporal. All variables define the data space that comprise our decision instances. Numerical and temporal variables can be binned into categorical variables. This way, they can be directly handled by our underlying decision-map analysis based on the MCA technique (Sec. 2.3). When quantizing numerical variables, users can choose the number of bins to match the precision needed for their questions. Temporal variables can be binned with various time spans, *e.g.*, year, quarter, month or day. Quantizing a variable can be iteratively done until the binned variable meets the analyst's expectations and needs.

To each variable node, we add, as leafs, all values that the respective category can take. For numerical and temporal variables, we show statistics (minimum, maximum, average, and standard deviation) to inform users about the basic properties of these variables. For each value of a categorical variable, we show a bar scaled by the number of times this value was taken in the dataset. The bars form together a histogram of the variable (see Figs. 1 and 6).

3.2.2 Analyzing Categorical Data

As outlined above, we reduce a decision model ontology to a categorical-only dataset. We next use Multiple Correspondence Analysis (MCA) to analyze this data. MCA treats rows (instances) and columns (variables) identically and thus allows analyzing both instances and variables. Intuitively, MCA can be seen as a weighted principal component analysis on either rows or columns [23].

Performing MCA can be done by constructing either an indicator matrix or a Burt matrix from the dataset [23]. In the indicator matrix I, we have a column for each possible value, with a position of 1 on rows (instances) that select this value. For any row, only one value of a particular variable can be set to one. The indicator matrix allows for analysis of both instances (rows) and variables (columns). However, the size of I is given by the number of instances, which can be very large. In our car insurance example, we have 100K decisions. As outlined in Sec. 2.2, this number can easily go into the millions. To support interactive real-time visual analysis, we thus base our framework on the Burt matrix, which is defined as

$$B = I^\top \times I \tag{1}$$

B is a symmetric square matrix whose rows and columns are the variable values. As a result, the size of B is bound by the number of *distinct* variable values. In our context, this number is several orders of magnitude smaller than the number of decisions. For example, in our car insurance scenario, the decision model contains 23 variables with a total of 158 distinct values; in contrast, we have 100K decisions.

MCA delivers three pieces of information for further analysis:

1. The *projections* of the original data points on the factors f_i, or eigenvectors, of the Burt matrix. These are sorted by explained variance, *i.e.* f_1 explains most of the variance, f_2 explains second most of the variance, and so on.
2. The amount of *variance* explained by each factor f_i.
3. The *contribution* of each variable to a factor, or how much of the variance of a given factor is explained by a given variable.

3.2.3 Decision Map

The *Decision Map* is a dynamic map analogous to the one presented by Zizi *et al.* [49]. However, while Zizi *et al.* assign space to areas in the map based on instances, *Decision Map* maps concepts.

The *Decision Map* follows the classical scatterplot technique used for MDS (Fig. 1B): We take the two factors f_1 and f_2 along which the data has most variance, and plot all variable value projections, or factor scores, along f_1 and f_2. Next, we map f_1 and f_2 to the x and y axes of our 2D scatterplot respectively. Similarity between variable values is show by proximity between 2D plot points.

MCA, like similar techniques such as MDS and PCA, tries to make the projection (2D) distance mirror the distance in the original data space. For the latter, we use the chi-square distance metric. This metric is based on relative frequencies of variables, adjusted by the contribution of a variable to the average instance. Each of the resulting factors is a combination of the variables used in the analysis. When most of the variance is explained by the first two factors f_1 and f_2, MCA is good for exposing the original data structure: Proximity between projected variable values means that those values are correlated. Additionally, contributions of each variable can be calculated, to explain how much the x and y plot axes are determined by a certain variable.

However, the complexity of the chi-square distance metric makes the interpretation of the x and y plot axes hard for business analysts. As such, we leave out these axes, and use the 2D scatterplot points as sites to create a Voronoi diagram. Due to the strong relation with ontology concepts, we name the resulting Voronoi cells *concept islands*. A concept island containing one concept represents all decisions that have this concept. Islands with more than one concept represent a set of decisions that have *all* these concepts.

As mentioned, each factor f_i explains part of the variance in the data and is a combination of the original variables. To make the *Decision Map* easier to read, we add three bar plots to it (for f_1, f_2, and for all factors $f_{i>2}$). Bars in the first two plots show the contribution of each variable to the x (f_1) and y (f_2) axes respectively. Bars in the third plot show the contributions of all variables which have *not* been captured by the plot, *i.e.* contributions captured by the factors $f_{i>2}$. Contributions in each plot, *i.e.* bar lengths, are sorted decreasingly, so we can locate the most important variables that map to the x and y axes or which are not captured by the 2D plot at all.

3.2.4 Color Mapping

The MCA visualization in [6] uses a categorical color map to show the identity of the mapped variables. The high number of colors in the view make it hard to interpret. In contrast, we use a two-color map. This two advantages. First, it allows us to show the relation between input and output concepts. More generally, we can now see dependent *vs* independent variables, where the user chooses which variables are of which type. Secondly, we avoid the color mapping problems that occur when color maps variable identity and when the number of variables is high (*i.e.* 10 or more, as noted in [6]).

The *Decision Map* in Fig. 4 illustrates the above. We see how the three values (eligible, manual and ineligible) of the decision concept *eligibility* (green concept islands) relate with respect to values of input concepts (blue concept islands).

3.2.5 Interaction

We support real-time exploration and filtering of decisions by several interaction techniques. First, we use the *DataSpace tree* to select the variables of interest, by (un)checking their respective checkboxes in this view (Fig. 1A). When variables are added or removed, MCA is performed anew on the selected variables and the *Decision Map* is updated. Given the efficient computation of the Burt matrix (Sec. 3.2.2), this process works in real-time even for large datasets. The implied

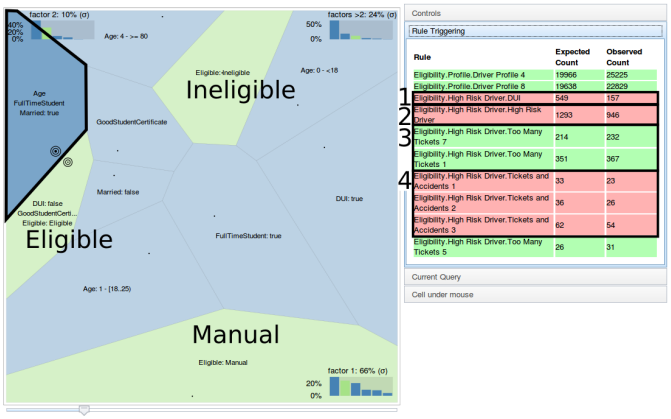


Fig. 4. Rule Triggering view for selections of decisions. The selected cell shows people between 25 and 80 who are maried and no longer full-time student. These people tend to get less often marked as high-risk drivers and are less often caught for driving under influence.

way of working is simple: An analyst starts with a selection of variables of interest determined by *e.g.* textual search (Sec. 3.1). From this selection, he next iteratively removes variables which do not contribute to the structure of the data (*e.g.* show up having small contribution in the *Decision Map* bar charts). This yields a *Decision Map* where real data correlations become more visible.

Secondly, we add interaction to the contribution bar charts to enhance understanding of the *Decision Map*. When a bar, representing a variable, is hovered over in one bar chart, bars in the other two charts representing the same variable are highlighted. This helps users to see how much a given variable is explained along the x and y axes, and also how much is not explained by the plot at all. At the same time, all concept islands that belong to the hovered variable are colored use a gradient color scheme based on ColorBrewer [24].

Thirdly, we provide a *merge slider*, which merges concept islands that are closer than the 2D distance given by the slider value. Merging results in new concept islands that represent not one value, but a set of correlated values. When a decision concept is merged with an input concept, *e.g.*, the *eligible* concept island in Fig. 4, the concept island is colored green as well. Merging helps users to segment the decision instances based on correlated properties.

Finally, each concept island can be hovered and brushed. At hovering, a detail panel on the right is updated to show the full name of this island. This way, we can show partial labels or no labels for small concept islands and concept islands that contain multiple values in the *Decision Map* to avoid clutter. Brushing implements dynamic queries [7] on the decisions by selecting one or multiple concept islands. By default, all decisions are shown in the decision table. When brushing concept islands, decisions are filtered based on the values represented by the brushed concept islands. This allows for creating selections of decisions (**T1**) based on correlations found in the data (*e.g.* find mid-aged and married people). Brushing allows discovering correspondences between the decision instances ($\mathbf{DS_2}$) and the decision logic ($\mathbf{DS_3}$). This is in contrast to most brushing techniques which provide detail-on-demand or show the same data in different (linked) views.

3.2.6 The Rule Trigger View

The *Decision Map* helps exploring relations between concepts ($\mathbf{DS_1}$) using the instances ($\mathbf{DS_2}$) of the taken decisions. So far, we only explored the insight that could be extracted by MCA on our decision model. Correlations between input concepts are, however, dictated by reality, *e.g.*, students tend to be younger on average; and mid-aged people have a higher chance of being married. Correlations between decision concepts and input concepts, on the other hand, result from the executed business logic given by the production rules. This distinction pertains precisely to our core question (Sec. 2): How can we examine how the reality (as captured by the decision model) *diverges* from the reality encoded by the actual decisions taken?

As explained, the *Decision Map* shows this distinction using a two-color mapping. We further analyze this distinction using the decision execution trace information available ($\mathbf{DS_3}$). Each decision is the result of production rules that triggered for a given input. For a random decision population, we can expect that each rule is triggered in the same proportion as for the overall population. Thus, selecting a particular decision subset and comparing the expected *vs* observed trigger counts for this selection shows us if the decision logic of this rule is either over- or underrepresented for this input population segment. For this, we sort the rules triggered for a decision subset by the absolute difference between the expected count and the observed count. Combining the correlation between properties with over- or underrepresented logic leads to a better understanding of

1. How a set of instances impacts the overall behavior of a decision.
2. How a particular formulation of rules leads to a unexpected behavior for a given set of instances.

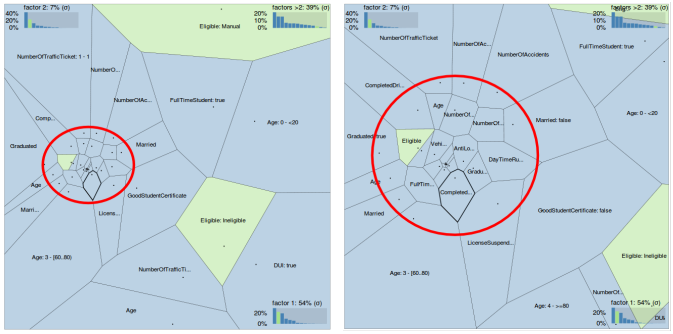
For this, we use a new view: The *Rule Trigger view* (Figs. 1C and 4). This view is a table showing the rules triggered for a selection of decisions. From left to right, the table columns show the unique rule identifier, expected trigger count, and actual trigger count for the selected decisions. The table background is colored red when the expected count is below the actual count, and green when the expected count is above the actual count. In Fig. 4, the selected concept island represents people between 25 and 80 who are married and no longer full-time student. From the corresponding *Rule Trigger view*, we learn that these people are less often caught for driving under influence (Fig. 4(1)) and get less often marked as high-risk drivers (Fig. 4(2)). They do get tickets (Fig. 4(3)), but this occurs less often in combination with accidents (Fig. 4(4)). As the selected island is close to the *eligibility* cell, this confirms the expectation (**T2,T3**) that people who are eligible for car insurance are indeed people with reasonable driving habits. We also considered showing rules not triggered for a selection. However, this creates visual clutter. For instance, when the selection contains only females, all rules that only apply to males will show up in the view.

3.3 Visualization refinements

We further found two issues related to using Voronoi cells to display concept islands. First, an informal user study with EDM domain experts at IBM showed that users find these cells confusing at first. For instance, users tried to interpret cell sizes, which have no direct semantics. We solved this issue by explaining that point proximity is the most important when looking at the plot. Next, we explained that a cell size relates to its position with respect to other points. That is, when a cell is small, the point is a center of a cluster, thus has many correlated values; when a cell is large, it is either at the border of a cluster or the value is an outlier. This explanation, and walking through a simple ten-minute training scenario, greatly helped users to learn how the plot should be used. However, more extensive user studies should point out if better visual encodings could be created instead of our Voronoi plot.

Secondly, we noticed that MCA tends to project most points close to the projection origin. These values reflect the properties of the *average* decision instance. Business-wise, these are initially the most important instances: Most decision instances have these properties and these have thus a large impact on business performance. Values that project as outliers are instance properties that occur less often, thus have likely a lesser business performance impact. Such outlier properties can be interesting in a second analysis pass, as they could suggest new business opportunities. Thus, visualization should first focus on properties of the average insurance requester. However, such properties get assigned the least space in the MCA plot.

We address the crowding issue by applying a transformation to the projected MCA values as follows. First, we scale both factors f_1 and f_2 to $[0..1]$. Next, we calculate the variance and the mean value for the values of the scaled factors. These values are then used to configure two scaling functions, one for each factor, that are centered around the



(a) (b)

Fig. 5. (a) Central clutter in MCA plots. (b) The spread transformation gives significantly more space to the values in the marked area (red circle) while keeping the overall plot structure intact.

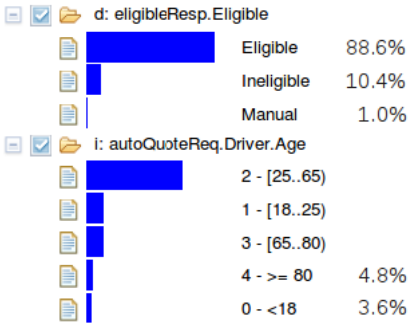


Fig. 6. DataSpace showing the *age* and *eligibility* variables.

mean. An uniform distribution on the $[0..1]$ domain has a variance of $1/12$. We use this observation by assuming that, when the variance is under $1/12$, the distribution is close to a normal distribution. Next, the scaling function behaves as a cumulative density function (CDF) for a normal distribution for variances below $1/12$ and as the inverse for variances above $1/12$. Overall, this scaling creates more space for clustered values while keeping the overall structure intact (Fig. 5), because both the CDF and its inverse are monotonically increasing.

4 Scenario: Car Insurance

We illustrate our DEL toolset with the analysis of a business process from the car insurance industry with the two decision models introduced in Sec. 2. The input instances have been synthesized to avoid confidentiality and privacy concerns. The ontology on which the decision models operate consists of an *AutoQuoteRequest*. This concept contains information about a driver, the car for which insurance is requested, and the insurance type.

The risk management decision logic (Fig. 2, $\mathbf{DM_1}$) has 22 rules. These do various checks on applicants, such as age checks and checks for high-risk driving indicators. The pricing decision logic (Fig. 2, $\mathbf{DM_2}$) has 144 rules. A set of decision tables determine the insurance base premium based on the type of the requested insurance and car value. Extra rules determine which discounts apply for a request, *e.g.*, anti-lock brakes and experienced-driver discounts. Yet other rules determine if certain surcharges apply, such as the old-vehicle surcharge. A final rules group model region-specific insurance policies. These either override global policies or encode additional discounts and surcharges that only apply in particular regions.

We next illustrate the usage of DEL by two user stories. In both cases, users are business analysts involved in deciding car insurance quotations. In the first story (Sec. 4.1), the analyst discovers that less people are eligible and wants to find out why. In the second story (Sec. 4.2), the analyst discovers that expensive cars do not get significant higher quotes.

4.1 Story 1: Why fewer than expected people are eligible

Using the *DataSpace tree* (Fig. 1A), the analyst examines the *eligible* decision concept. He sees that 88.6% of the requests are eligible, 10.4% are ineligible, and 1% are manually processed (Fig. 6). This is unexpected, because the analyst designed this decision model to have an eligibility rate between 90% and 95%. Thus, the analyst first learns that his model does not meet the expected performance (**T3**).

Using his knowledge about the decision model and business domain, the analyst now states that people below 18 and people above 80 are ineligible due to legislation and business policies. He now wants to check this expectation against the actual age distribution of drivers. For this, he quantizes the numerical *age* variable into five categories, two for the lower and upper ranges of ineligible people, and three for potentially eligible people: <18: Not eligible, $[18..25)$: Youth, $[25..65)$: Adults, $[65..80)$: Elderly, and ≥ 80: Not eligible.

The quantized *age* concept appears in the *DataSpace tree* under categorical concepts (Fig. 6). The lower and upper age category percentages lead the analysts to deduct that these two groups account for about 85% of the ineligible requests. Thus, 15% of all ineligible applicants (1.6% of the total) are so because of other reasons than age. This is a first clue that there is some room for decision improvement, *i.e.*, maximizing the number of insurances sold. For this, we need to find out why people are ineligible for other reasons than their age (**T4**).

The analyst is next interested in the correlation between driver attributes and the decision (**T2**). He thus selects all driver attributes and the *eligibility* decision attribute in the *DataSpace tree*. The resulting plot (Fig. 7a) shows some structure, which is hard to examine due to clutter. Values above the topleft-bottomright diagonal represent ineligible people. As expected, the $age < 18$ and $age > 80$ values fall into this area. Values below this diagonal are related to eligible or manually-processed requests.

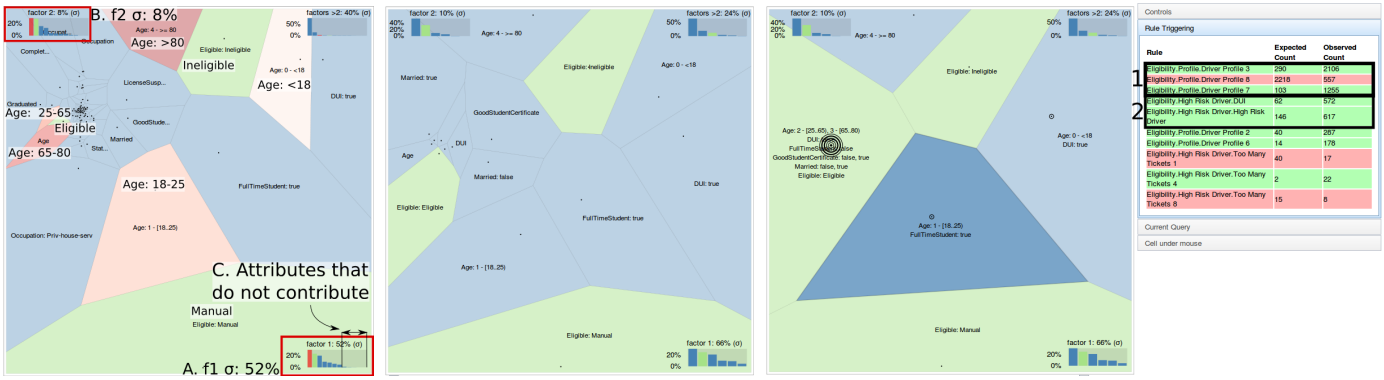
From the contribution plots (Fig. 7a(A,B)), the analyst learns that only 60% of the variance is explained by the first two factors. Also, he finds that there are several attributes that do not contribute to the data structure (**T2**). These are the attributes in the bottom-right and top-left contribution plots that are on the far right of these plots (Fig. 7a C). Among these, we have *state* and *gender*. This tells the analyst that state and gender do not influence the decision.

Next, the analyst iteratively removes these unimportant attributes until all remaining attributes contribute at least 5% to f_1 (x axis). He now obtains a simplified map (Fig. 7b), with the attributes *age*, *eligibility*, *full-time student*, *married*, *DUI* (driving under influence), and *good student* certificate. This map explains 76% of the variance and still shows the earlier diagonal structure. This confirms the earlier finding that the removed attributes do not contribute to the data structure.

To further simplify the view, the analyst uses the merge slider to group correlated values in single concept islands. Left in the resulting *Decision Map* (Fig. 7c), he sees the concept island for *eligible* instances. The values of the two most important input attributes are $age : [25..65, 65..80]$ and $fulltimestudent : false$. Hence, the expectation that eligible people are mainly working adults is confirmed (**T3**). At the top in Fig. 7c, the analyst sees that ineligible people are correlated to $age \geq 80$. For instances where $age < 18$, the most important correlated value is *DUI*.

In the center of the *Decision Map* (Fig. 7c), the analyst finds a concept island for students between 18 and 25. This island is centered between the three decision values (*eligible*, *ineligible*, *manual processing*), showing that these decisions are about equally spread among students. He now sees that students are a group where some optimization may be possible, as these are not strongly correlated with any of the decision outcomes (**T4**). To find out more about students, the analyst selects the center cell to filter decisions and update the *Rule Trigger view*. This view now shows only decisions for instances that have $age \in [18..25)$ and $fulltime\ student = true$ (5716 decisions).

Knowing the decision model, the analyst notes that the first three rules in the *Rule Trigger View* (Fig. 7c(1)) come from a decision table. For the definition of these particular rules, the analyst turns to his rule authoring environment, IBM Operational Decision Manager (ODM) in this case [25]. In ODM, he finds the decision table con-



(a) (b) (c)

Fig. 7. Interactive analysis of variables of interest: (a) The analyst has selected all available attributes related to drivers and clicked the plot *age* bar to see where the various age groups are in the plot. (b) The analyst simplified the plot by removing all variables that did not contribute to the data structure. (c) The analyst merged values and selected the 'students' concept island to examine rule triggering for this set of decision instances.

	Gender	Age	State of Residence	Set Eligibility	
				Eligible?	Message
1	female	< 16		Ineligible	Sorry, you are too young to q...
2		[16 25[	CA	Manual	Females between 16 and 25 i...
3			*Otherwise*	Eligible	Congratulations! Your applica...
4		≥ 25		Eligible	Congratulations! Your applica...
5	male	< 18		Ineligible	Sorry, you are too young to q...
6		[18 21[	CA	Manual	Males between 18 and 21 in C...
7			*Otherwise*	Eligible	Congratulations! Your applica...
8		≥ 21		Eligible	Congratulations! Your applica...

Fig. 8. Decision table for initial eligibility status based on age.

taining the rules he just found in the *Rule Trigger View* (Fig. 8). From their definitions, he sees that these rules do a basic separation mostly based on age. In this decision table, the analyst sees that rules 3 and 7 affect people between 16 and 25 not living in CA, and rule 8 represents males above 21. Given that he is looking at students, it is not much of a surprise that rule 3 and 7 are overrepresented, while rule 8 is underrepresented for this selection of decisions (**T3**).

The two rules from the High Risk Driver package (Fig. 7c(2)) reveal an interesting fact about the working of the decision model. Both the *DUI* and the *high-risk driver* rule were not specifically written with students in mind. However, students trigger these rules more than the analyst expects, given that they were not written for this group *in particular*. The analyst knows that, when an insurance request is flagged as being from a high risk driver, the decision model always rejects it. However, he also knows that business policies state that extra effort should be done to make students customer. Using the above analysis, the analyst decides to add a rule to the decision model which marks high-risk students for manual processing. This way, such students will be analyzed by salespeople (rather than automatically rejected), and thus have a higher chance to become customers (**T4**).

Summarizing, the analyst first found a hint that there is space for improvement in the *DataSpace tree*. Next, he used the *Decision Map* to find clusters of correlated values based on the accumulated decision instances. Finally, he uses external contextual information about business policies with respect to retaining students to find ways to improve the decision model.

4.2 Story 2: Why expensive cars get low quotes

The KPI of the insurance decision model (Fig. 2, $\mathbf{DM_2}$) is computed as follows. First, the sum Q of all calculated insurance quotes is computed. Next, for each insurance, two probabilities ψ_1 and ψ_2 are drawn from a car-accident and a claim-value distribution respectively. ψ_1 and ψ_2 are combined to calculate the probable insurance loss, which is then subtracted from Q. Monitoring this KPI over time shows *whether* the business is profitable, but not *why*. Also, the KPI is less precise than it could be, given that the car-accident and claim-value distributions do not consider the specifics of certain insurance instances.

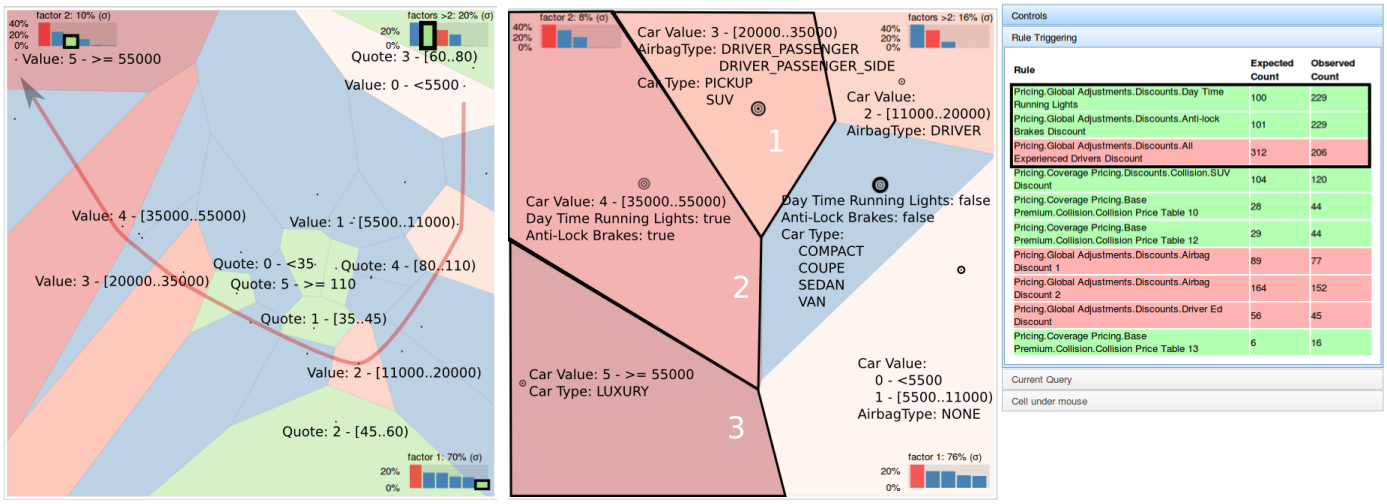
To increase insight in the business performance, the analyst decides to analyze the insurance quotes. For this, he quantizes the numerical attribute *vehicle.value* and *quote*. The car value is quantized in the same way as it is done in the production rules that determine the quote base premium (< 5500, 5500..11000, 11000..20000, 20000..35000, 35000..55000, ≥ 55000). Next, the analyst assumes that quotes are correlated to car values. Hence, he quantizes the *quote* variable in proportional ranges to the quantization of the *value* variable.

Next, the analyst selects properties of drivers, cars, and the *quote* variable. The contribution plots show long tails of non contributing attributes. Brushing next shows that those are driver properties. This is not surprising, as the analyst assumes that car properties, especially car value, are the most important factors for the quote. To clean up the plot, he now iteratively removes such unimportant variables, as explained for story 1 (Sec. 4.1). After removing the driver attributes from the plot, the analyst finds, to his surprise (**T3**), that there is no strong correlation between car value and quote. To learn how car values are distributed in the *Decision Map*, the analyst now clicks the 'Car Value' bar in one of the contribution plots. The gradient coloring shows its value for ordered attributes, such as age and car value in this particular case. As visible in Fig. 9a, a clockwise gradient (marked with the red arrow) shows low-priced cars placed top right; increasing car values in-between; and expensive cars top-left. Once this pattern is clear, the analyst now understands how spatial areas relate to car values. The *quote* attribute values, on the other hand, are almost all centered, and do not show a clear vertical or horizontal distribution. This finding is enforced by the contribution plots which show that the *quote* attribute (green bar) does not contribute much to either f_1 (x axis) or f_2 (y axis).

The analyst now removes the *quote* variable from the plot and performs merging in order to get clusters of related values (Fig. 9b). He notices that car options such as all-side airbags (concept island 1), anti lock brakes, and daytime running lights (concept island 2) correlate with the more expensive cars. At this point, he selects the concept islands related to expensive cars (islands 1, 2 and 3 in Fig. 9b). Now, he finds two interesting things in the *Rule Triggering view*: First, the rule that gives a discount for experienced drivers is underrepresented. For this selection, it is expected that this rule should trigger 312 times, but it is only triggered 206 times. Thus, expensive cars are less often driven by drivers that are considered experienced by the business logic. Secondly, the discount rules for daytime running lights and anti-lock brakes are overrepresented. The daytime running light discount is triggered 229 times times instead of the expected 100; and the anti-lock brakes discount is triggered 229 times instead of the expected 101. When the analyst selects each of these three concept island individually, he finds two more overrepresented discounts: In concept island 3, the passenger-and-driver airbag discount rule is overrepresented; in concept island 1, the SUV discount rule is overrepresented.

From all the above, the analyst has finally now learned that:

1. The car value is, unexpectedly, not strongly correlated to the



(a) (b)

Fig. 9. Car values correlated to quotes in an unexpected way. (a) The highlighted values show a clockwise trend for car values with low valued cars top right and high valued cars top left. (b) After removing the quote and performing some merging the same trend is still visible (though counter clockwise). Additionally, The triggered rules view reveal some cumulating discounts and an underrepresentation of experienced drivers.

quote (**T2, T3**).

2. Expensive cars are less likely driven by experienced drivers (**T3**).
3. While the base quote calculation takes into account car values, expensive cars still get too low quotes due to an accumulation of discounts (**T3**).

As a result, the analyst has discovered a potential revenue leak in the business logic, which could be compensated for instance by setting up a 'luxury car' surcharge rule and using different calculations for the base quote (**T4**). Of course, this decision is subject to various checks with actuary departments and requires further investigations before being implemented. Nonetheless, DEL has shown here its potential for insight formation.

5 Implementation

DEL is implemented as an extension for IBM Operational Decision Manager [25]. Its input data is gathered from ODM's decision warehouse. We extract the data from this warehouse in a separate Apache Solr search server [4]. This decouples the load caused by analysis from production systems. It also allows data to be indexed and stored in a suitable format for efficient querying (Sec. 3.1). With this architecture, MCA can be performed in real time for data sets with tens of thousands of items. Quantizing numerical variables, a task that goes over all decisions, takes more time when the number of decisions grows. However, this is normally done only at the beginning of the analysis process and hence does not have a significant performance impact.

DEL consists of a back end, which is implemented using Java technologies, and a web based front end. The front-end is implemented using the Dojo Toolkit [10] for the overall layout and standard widgets. For the *Decision Map*, we used D3.js [5] which provides the needed flexibility for custom widgets and interactions tightly bound to the data at hand. DEL was tested on a standard dual-core 2.5 Ghz machine with 4GB of RAM. On this configuration, DEL can easily handle over 100K decision instances in real-time.

DEL was evaluated by IBM ODM's product managers and chief architects. As a preliminary validation of our work, we quote below one of the ODM chief architects:

"The Decision Exploration Lab would be a key asset for the ODM product to be able to assist our users with the right level of tooling and targeted analytics. From my point of view, in this proof of concept, the algorithms are sound and the visualization is great. We will need more use case investigations, refinement on the visual design and some task-based approaches to adapt the ease of use to the level of understanding of our users. But I'm sure this mid-point exists, all the more as our users, in these times of big data and business analytics, are becoming more advanced and less intimidated by analytics in general."

6 Conclusion

We have presented the Decision Exploration Lab (DEL), a visual analytics solution designed to address prevalent issues in the area of Operational Decision Management (ODM). Our solution integrates production rule execution traces, statistical analysis of decision instances, and interactive visual analysis into a decision exploration system. We demonstrated how the tool can be used to verify expected and find unexpected functioning of decision models, using a scenario from the car insurance industry.

The industrial relevance of the problems addressed by DEL was expressed by one of the IBM ODM chief architects as follows: "A significant part of the ODM customers implements a feedback loop to improve the the logic of their automated decisions from the actual outcomes of decisions. They gain insight in the business efficiency of these decisions by monitoring KPIs and Business Intelligence approaches. But when it comes to figuring out what concrete parts of a decision model have to be improved or rethought to meet a particular goal, they can only rely on their knowledge."

The originality of our approach is that we provide a toolset that supports combined analysis of rule execution traces and decision instances. This allows business analysts to explore decision models in the light of the accumulated facts about this model in the form of decision instances. As a result, analysts can verify if the decision model and the reality are diverging, and if so, understand the underlying reasons and take corrective and perfective measures.

Understanding the functioning of decision models is a challenging topic and much more work is to be done. Both the ontology and the production rules of a decision model contain a wealth of information that can be included in the visual analysis process. In the future, we will explore how this information can be integrated and used to present richer and more precise information about the decision model. We already performed informal user studies with internal domain experts to verify the soundness of our approach. More controlled user studies will be performed to evaluate the overall effectiveness of our approach. Based on the lessons learned, we are now working with the IBM ODM product team to devise future directions of DEL in this area.

REFERENCES

[1] S. Afzal, R. Maciejewski, and D. S. Ebert. Visual analytics decision support environment for epidemic modeling and response evaluation. In *Proc. IEEE VAST*, pages 191–200, 2011.

[2] S. Al-Hajj, I. Pike, B. Riecke, and B. Fisher. Visual analytics for public health: Supporting knowledge construction and decision-making. In *Proc. HICSS*, pages 2416–2423, 2013.

[3] B. Alsallakh, E. Gröller, S. Miksch, and M. Suntinger. Contingency wheel: Visual analysis of large contingency tables. In *Proc. EuroVA*, pages 53–56, 2011.

[4] Apache Software Foundation. Apache Solr, 2013. lucene.apache.org/solr.

[5] M. Bostock. D3.js data-driven documents. www.d3js.org, 2013.

[6] B. Broeksema, A. Telea, and T. Baudel. Visual analysis of multidimensional categorical datasets. *Submitted to CGF*, 2013.

[7] S. K. Card, J. D. Mackinlay, and B. Shneiderman. *Readings in Information Visualization: Using Vision to Think*. Academic Press, 1999.

[8] B. Cornelissen, A. Zaidman, A. van Deursen, L. Moonen, and R. Koschke. A systematic survey of program comprehension through dynamic analysis. *IEEE TSE*, 35(5):684–702, 2009.

[9] S. Diehl. *Software Visualization: Visualizing the Structure, Behaviour, and Evolution of Software*. Springer, 2007.

[10] Dojo Foundation. Dojo Toolkit. dojotoolkit.org, 2013.

[11] F. N. E. and R. Schwitter. Specifying logic programs in controlled natural language. In *Proc. Computational Logic for Natural Language Processing*, pages 3–5, 1995.

[12] N. E. Fenton. *Software Metrics: A Rigorous and Practical Approach*. Thomson Computer Press, 2^{nd} edition, 1996.

[13] S. J. Fernstad and J. Johansson. A task based performance evaluation of visualization approaches for categorical data analysis. In *Proc. Information Visualisation*, pages 80–89. IEEE, 2011.

[14] FICO, Inc. Blaze Advisor Business Rules Management. www.fico.com/en/Products/DMTools/Pages/FICO-Blaze-Advisor-System.aspx, 2013.

[15] I. Fodor. A survey of dimension reduction techniques. Technical report, Center for Appl. Sci. Comp., LLNL, 2002.

[16] C. Forgy. RETE: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, 19(1):17–37, 1982.

[17] M. Friendly. Mosaic displays for multi-way contingency tables. *JSTOR*, 89(425):190–200, 1994.

[18] M. Friendly. Extending mosaic displays: Marginal, conditional, and partial views of categorical data. *J. Comput. Graph. Stat.*, 8:373–395, 1999.

[19] M. Friendly. Visualizing categorical data: Data, stories, and pictures. In *Proc. SAS User Group Conf.*, 2000.

[20] K. Gabriel. The biplot graphic display of matrices with application to principal component analysis. *Biometrika*, 58(3):453–467, 1971.

[21] T. M. Green, W. Ribarsky, and B. Fisher. Visual analytics for complex concepts using a human cognition model. In *Proc. IEEE VAST*, pages 91–98, 2008.

[22] M. J. Greenacre. *Correspondence analysis in practice*. CRC Press, 2007.

[23] M. J. Greenacre. *Biplots in Practice*. Fundación BBVA, 2010.

[24] M. A. Harrower and C. A. Brewer. Colorbrewer: An online tool for selecting color schemes for maps. *Cartogr. J.*, pages 27–37, 2003.

[25] IBM. Operational Decision Manager. www-01.ibm.com/software/decision-management/operational-decision-management/websphere-operational-decision-management, 2013.

[26] P. Jackson. *Introduction to Expert Systems*. Addison-Wesley, 2^{nd} edition, 1990.

[27] JBoss. Drools business logic integration platform. www.jboss.org/drools, 2013.

[28] B. Johnson and B. Shneiderman. Treemaps: a space-filling approach to the visualization of hierarchical information structures. In *Proc. Infovis*, pages 284–291. IEEE, 1991.

[29] A. Katifori, C. Halatsis, G. Lepouras, C. Vassilakis, and E. Giannopoulou. Ontology visualization methods: a survey. *ACM Comput. Surv.*, 39(4), 2007.

[30] R. Ko, S. Lee, and E. Lee. Business process management (bpm) standards: a survey. *J. Business Process Management*, 15(5):744–791, 2009.

[31] E. Kolatch and B. Weinstein. Cattrees: Dynamic visualization of categorical data using treemaps, 2013. www.cs.umd.edu/class/spring2001/cmsc838b/Project/Kolatch_Weinstein.

[32] R. Kosara, F. Bendix, and H. Hauser. Parallel sets: Interactive exploration and visual analysis of categorical data. *IEEE TVCG*, 12(4):558–568, 2006.

[33] M. Lanza, R. Marinescu, and S. Ducasse. *Object-Oriented Metrics in Practice*. Springer, 2005.

[34] C. Leondes. *Expert systems: the technology of knowledge management and decision making for the* 21^{st} *century*. Academic Press, 2002.

[35] R. Lu and S. Sadiq. A survey of comparative business process modeling approaches. In *Business Information Systems*, pages 82–94. Springer, 2007.

[36] J. Maletic, A. Marcus, and M. Collard. A task oriented view of software visualization. In *Proc. IEEE VISSOFT*, page 3240, 2002.

[37] M. Migut and M. Worring. Visual exploration of classification models for risk assessment. In *Proc. IEEE VAST*, pages 11–18, 2010.

[38] D. Miranker. TREAT: A better match algorithm for ai production systems. Technical report, AI-87-58, Univ. of Texas, Austin, 1987.

[39] F. Niedermann and H. Schwarz. Deep business optimization: Making business process optimization theory work in practice. In *Enterprise, Business-Process and Information Systems Modeling*, pages 88–102. Springer, 2011.

[40] Object Management Group. Business Process Model and Notation. www.bpmn.org, 2013.

[41] S. J. Russell, P. Norvig, J. F. Candy, J. M. Malik, and D. D. Edwards. *Artificial intelligence: A modern approach*. Prentice Hall, 1996.

[42] A. Savikhin, H. Lam, B. Fisher, and D. Ebert. An experimental study of financial portfolio selection with visual analytics for decision support. In *Proc. HICSS*, pages 1–10, 2011.

[43] A. Savikhin, R. Maciejewski, and D. S. Ebert. Applied visual analytics for economic decision-making. In *Proc. IEEE VAST*, pages 107–114, 2008.

[44] M. Scaife and Y. Rogers. External cognition: how do graphical representations work? *Int. J. Hum.-Comput. Stud.*, 45(2):185–213, 1996.

[45] J. Taylor and N. Raden. *Smart Enough Systems*. Prentice Hall, 2004.

[46] M. Tenenhaus and F. Young. An analysis and synthesis of multiple correspondence analysis, optimal scaling, dual scaling, homogeneity analysis and other methods for quantifying categorical multivariate data. *Psychometrika*, 50(1):91–119, 1985.

[47] X. Wang, W. Dou, T. Butkiewicz, E. Bier, and W. Ribarsky. A two-stage framework for designing visual analytics system in organizational environments. In *Proc. IEEE VAST*, pages 251–260, 2011.

[48] A. Wlodyka, R. Mlynarski, G. Ilczuk, E. Pilat, and W. Kargul. Visualization of decision rules - from the cardiologists point of view. In *Proc. Computers in Cardiology*, pages 645–648, 2008.

[49] M. Zizi and M. Beaudouin-Lafon. Accessing hyperdocuments through interactive dynamic maps. In *Proc. ACM ECHT*, pages 126–135, 1994.