NEW GENERATION PLATFORM FOR MULTI-CRITERIA QUALITATIVE DECISION MODELLING WITH METHOD DEX

Nejc Trdin

Doctoral Dissertation
Jožef Stefan International Postgraduate School
Ljubljana, Slovenia

Supervisor: Prof. Dr. Marko Bohanec, Jožef Stefan Institute, Ljubljana, Slovenia

Evaluation Board:
Prof. Dr. Bojan Cestnik, Chair, Temida d. o. o., Ljubljana, Slovenia
Asst. Prof. Dr. Martin Žnidaršič, Member, Jožef Stefan Institute, Ljubljana, Slovenia
Prof. Dr. Blaž Zupan, Member, Faculty of Computer and Information Science, University of Ljubljana, Ljubljana, Slovenia

MEDNARODNA PODIPLOMSKA ŠOLA JOŽEFA STEFANA
JOŽEF STEFAN INTERNATIONAL POSTGRADUATE SCHOOL

Nejc Trdin

NEW GENERATION PLATFORM FOR MULTI-CRITERIA QUALITATIVE DECISION MODELLING WITH METHOD DEX

Doctoral Dissertation

PLATFORMA NOVE GENERACIJE ZA VEČKRITERIJSKO KVALITATIVNO MODELIRANJE Z METODO DEX

Doktorska disertacija

Supervisor: Prof. Dr. Marko Bohanec

Ljubljana, Slovenia, October 2015

To my Parents

Acknowledgments

I would like to express my gratitude to all my friends and colleagues who contributed to this thesis, be it during discussion of my work over a cup of coffee, or during day-to-day topics over a glass of beer.

First and foremost, I would like to thank my supervisor Prof. Dr. Marko Bohanec. He was a great source of motivation and inspiration during my studies. He provided professional guidance, support and understanding even in the most difficult situations. His valuable expertise and his tendency for perfection made my work better with each meeting we had. I am especially grateful to him for not only being my supervisor, but also for being a friend. I will never forget the iced coffee in Anavyssos, tapas and beer in Málaga, and sushi in Paris, Barcelona, Frankfurt and Hamburg! Thank you Marko!

I wish to express my gratitude to the committee members, Prof. Dr. Bojan Cestnik, Asst. Prof. Dr. Martin Žnidaršič and Prof. Dr. Blaž Zupan, for their most valuable comments and remarks during my Seminar presentations.

My sincere thanks go to all members of the department of Knowledge Technologies and all other colleagues at the Jožef Stefan Institute. Thank you Vlado, Jovan, Darko, Jasmina, Jurica, Nikola, Hristijan and Martin, for all the lunches and breaks we enjoyed together. I would also like to express my gratitude to my office colleagues and friends, Biljana and Dragana. They both gave valuable advice during discussions about my work. With them by my side, my work environment was even more enjoyable.

My off-work friends, Zagi, Blaž, Jernej, Urh, Rok, Miha, Domen, Žiga, Goran, Jaka, Tadej, and Matej, showed me the true meaning of being a friend. Without them, my work would not be half as enjoyable, as it was. I would especially like to thank Zagi for reminding me what a best friend is. Also, I would like to thank Jernej for all the fruitful discussions we had over lunch, be it about computer science, math or small talk.

I am very grateful for the financial support of the Jožef Stefan Institute and the Slovenian Research Agency through Young Researchers Programme.

My sincere gratitude goes to my parents for all the financial support during my studies in Ljubljana, for moral support when I needed it, and for always being there for me. I would like to thank my sister Ajda for all the things she did for me — keeping me in touch with my family in Velenje, having conversations over lunch, keeping me up-to-date with gossip, and especially for being a sister.

Finally, I would like to thank my girlfriend Urška. She lifted me up, when I was down. She gave me inspiration when I was stuck, and saw no way out. She listened to my problems and nagging, and even after that kept a positive view on the situation. She was there for me at every moment. Thank you!

Abstract

DEX is a qualitative multi-criteria decision analysis method. It supports decision makers in making complex decisions based on multiple, possibly conflicting, attributes. The attributes in DEX have qualitative value scales and are structured hierarchically. The hierarchical topology allows for decomposition of the decision problem into simpler subproblems. In DEX, alternatives are described with qualitative values, taken from the corresponding hierarchy input attribute scales. The evaluation of alternatives is performed in a bottom-up way, utilizing aggregation functions, which are defined for every aggregated attribute in the form of decision rules. DEX has been used in numerous practical applications — from everyday decision problems to solving decision problems in the financial and ecological domains.

DEX's wide use in practice revealed the need to extend it in several directions. The extensions facilitate decision makers in more complex decision problem solving. The following five methodological extensions were identified: support for full hierarchies, introducing numeric attributes, using general aggregation functions, probabilistic and fuzzy aggregation of values, and relational models.

This thesis develops and extends DEX in the identified directions. Further, each extension is formalized in order to include it in the formalization of the extended method. Even more, the developed formalization allows for usage of all extensions at once — in one decision making problem. We call the extended method DEXx.

The formalized DEXx is implemented in a library, which supports basic DEX functions together with the identified formalized extensions. The library natively supports building DEXx models, with added extensions, and evaluating alternatives. Even more, the library supports new utilities, which enable the decision maker to further analyse and view the developed models.

DEXx is evaluated through four complex real-life decision making use-cases, using the DEXx implementation. The construction and evaluation of use-cases show the added benefits of the extensions to the decision making process. The use-cases are qualitative assessment of public e-portals, assessment of bank reputational risk, sustainability assessment of electric energy production technologies in Slovenia, and the model for water flows in agriculture. The models for use-cases are developed using only native DEXx framework — eliminating the need for additional programming or manual work.

Through the use-cases we show that identified extensions provide additional means for the decision maker during the decision making process — the extensions increase the expressive power of the methodology, which in turn facilitates solving of a larger class of decision problems.

Povzetek

DEX je kvalitativna večparametrska metoda za odločitveno analizo. Metoda podpira odločevalce pri sprejemanju kompleksnih odločitev, ki pogosto temeljijo na več pogosto nasprotujočih si atributih. Atributi v DEX-u imajo kvalitativne (simbolične) merske lestvice in so strukturirani hierarhično. Hierarhična topologija omogoča razgradnjo odločitvenega problema v preprostejše podprobleme. V DEX-u so alternative opisane s kvalitativnimi vrednostmi, vzetimi iz ustreznih merskih lestvic vhodnih atributov hierarhije. Vrednotenje alternativ poteka na način od spodaj-navzgor z uporabo funkcij združevanja, ki so opredeljene za vsak združen atribut v obliki odločitvenih pravil. DEX je bil v preteklosti uporabljen v številnih praktičnih primerih — od vsakdanjih problemov odločanja do reševanja odločitvenih problemov na finančnih in ekoloških področjih.

V praksi so se pokazale potrebe po razširitvi metode DEX v več smeri. Razširitve omogočajo odločevalcem sprejemanje odločitev v kompleksnejših odločitvenih problemih. Identificirali smo sledečih pet metodoloških razširitev: podpora za polne hierarhije, podpora za numerične atribute, uporaba splošnih funkcij združevanja, verjetnostno in mehko združevanje vrednosti in podpora za relacijske modele.

Ta teza razvije in razširi metodo DEX v identificirane smeri. Nadalje, vsaka razširitev je formalizirana, da se lahko vključi v formalizacijo razširjene metode. Še več, formalizacija je razvita tako, da se lahko vse razširitve uporabljajo hkrati — v enem odločitvenem problemu. Razširjeno metodo imenujemo DEXx.

Formalizirani DEXx je implementiran v knjižnici, ki podpira osnovno metodo DEX, skupaj z identificiranimi formaliziranimi razširitvami. Knjižnica v osnovi podpira gradnjo DEXx modelov z dodanimi razširitvami in vrednotenje alternativ. Še več, knjižnica podpira nove funkcionalnosti, ki omogočajo odločevalcu, da nadalje analizira in pregleduje razvite modele.

Metoda DEXx je ovrednotena skozi štiri kompleksne primere uporabe iz realnega življenja s pomočjo implementacije DEXx. Konstrukcija in vrednotenje primerov uporabe kažeta dodane koristi razširitev v procesu odločanja. Primeri uporabe so: kvalitativna ocena javnih e-portalov, ocena tveganja ugleda banke, ocena vzdržnosti elektroenergetskih tehnologij v Sloveniji in model za ocenjevanje tokov vode v kmetijstvu. Modeli za primere uporabe so razviti z uporabo knjižnice DEXx — to odpravlja potrebo po dodatnih programih ali ročnem delu.

Skozi primere uporabe smo pokazali, da predstavljene razširitve zagotavljajo dodatna sredstva odločevalcu v procesu odločanja — razširitve povečajo izrazno moč metodologije, ki posledično omogoča reševanje več vrst odločitvenih problemov.

Contents

List of Figures

List of Tables

List of Algorithms

Abbreviations

AHP	...	Analytic Hierarchy Process
CBR	...	Case-Based Reasoning
CBRR	...	Case-Based Rule Reasoning
DAG	...	Directed Acyclic Graph
DECMAK	...	DECision MAKer
DEX	...	Decision EXpert
DEXx	...	Decision EXpert X
DM	...	Decision Maker
DOCTUS	...	Knowledge-based expert system shell for evaluation of alternatives
DRSA	...	Dominance-based Rough Set Approach
DSS	...	Decision Support System
GUI	...	Graphical User Interface
HINT	...	Hierarchy INduction Tool
MACBETH	...	Measuring Attractiveness by Categorical Based Evaluation TecHnique
MAUT	...	Multi Attribute Utility Theory
MCDA	...	Multi-Criteria Decision Analysis
MCDM	...	Multi-Criteria Decision Modeling
ORCLASS	...	ORdinal CLASSification
PDF	...	Probability Density Function
PMF	...	Probability Mass Function
RBR	...	Rule-Based Reasoning
UTA	...	UTilitiés Additives, an outranking MCDM method
VDA	...	Verbal Decision Analysis
ZAPROS	...	Russian abbreviation for Closed Procedures near Reference Situations

Chapter 1

Introduction

Decision making and more specifically *decision support* is an area aimed at supporting area experts in making their decisions (Bouyssou, Marchant, Pirlot, Tsoukiàs, & Vincke, 2006; French, 1986; Figueira, Greco, & Ehrgott, 2005). Given some *decision problem* with *decision alternatives (options)*, the *decision maker (DM)* needs to select the *best* alternative, from the given set of alternatives. A *decision model* is constructed from *preferences* defined by the DM — in this thesis we refer to DM as the decision maker, possibly assisted by users, experts, stakeholders and analysts from the problem area. The developed decision model facilitates the DM in the alternative selection process. The model can also bring additional insight to the DM regarding the decision problem.

Generally, the approach is that the DM defines and constructs a decision support model using a selected methodology for a specific decision problem in accordance with their preferences. While constructing the model, the DM also provides their preferences regarding attributes, attribute values and/or given alternatives. The alternatives are evaluated using the developed model, and the best alternative is selected by the DM — after optional analysis of the model.

Multi-criteria decision making (MCDM) is a sub-field of decision making, specifically of operations research, where decision problems are composed of multiple, typically conflicting criteria. The field considers criteria that need to be evaluated in order to make and accept a final decision (Bouyssou et al., 2006; French, 1986; Figueira et al., 2005). This thesis is concerned with the special case of *qualitative* multi-criteria decision analysis methods. These methods are split into two groups, according to the way the knowledge acquisition (Boose, Bradshaw, Koszarek, & Shema, 1993; H.-L. Yang, 1995) from the decision maker is done: in the first group are the methods which are based on interactive questioning procedure for obtaining decision maker's preference; methods in the second group avoid the long lasting questioning procedures, they rather use DM's preference directly while building the decision model. Specifically, in this thesis we are concerned with method DEX, which belongs to the second group.

DEX (Decision EXpert) is a qualitative multi-attribute modeling method. DEX has its roots in fuzzy heuristics approaches (Efstathiou & Rajkovič, 1980), which were extended and implemented in the systems DECMAK (Bohanec & Rajkovič, 1983; Rajkovič, Bohanec, & Efstathiou, 1987) and DEX (Bohanec & Rajkovič, 1990). A DEX model enables *hierarchical* structuring of qualitative attributes — attributes are split into multiple *child (descendant)* attributes so that each child attribute represents a more basic concept in the decision problem. The structuring continues until all values for *input (leaf)* attributes can be derived easily from the alternative's properties. Input attributes are the ones that do not have any child attributes. On the other hand, *root* attributes are the ones that do not have any *parent* attributes. The attributes with child attributes are called *aggregated*

attributes. These attributes must have an aggregation function defined, in order to facilitate model evaluation. Since DEX is a qualitative MCDM method, the values in the *scale* of the attributes are *words (symbolic values)* rather than numbers. The aggregation of values is defined by *decision rules*, which can be interpreted as a collection of *if-then* rules. Acquisition of knowledge for defining aggregation functions in DEX is done by questioning the DM, what the value of output attribute would be, in case of the given input values. Alternatives in DEX are represented with tuples of values — a qualitative value is assigned to the corresponding input attribute. Evaluation of the alternatives in DEX is done in a *bottom-up* fashion, progressively computing aggregation functions, from inputs to the root attributes. The final evaluation of an alternative is represented by the values computed in the root attributes. DEX is described in more detail in Chapter 3.

DEX has been widely used in practice to support complex decision processes in health threats and crisis management (Žnidaršič, Bohanec, Lavrač, & Cestnik, 2009), the use of genetically modified crops (Bohanec et al., 2009, 2008; Bohanec & Žnidaršič, 2008; Žnidaršič, Bohanec, & Zupan, 2008), the evaluation of data mining work flows (Žnidaršič, Bohanec, & Trdin, 2012), the evaluation of public administration portals (Leben, Kunstelj, Bohanec, & Vintar, 2006), management of water sources (Trdin, Bohanec, & Janža, 2013; Trdin, 2011), the assessment of bank reputational risk (Bohanec, Aprile, Constante, Foti, & Trdin, 2013), the evaluation of stock options (Alić, Siering, & Bohanec, 2013), environmental decision making (Kuzmanovski, 2012; Kuzmanovski, Trajanov, Leprince, Džeroski, & Debeljak, 2015; Žnidaršič, Bohanec, & Zupan, 2006b) and many others (Bohanec, Rajkovič, Bratko, Zupan, & Žnidaršič, 2013).

The many uses of DEX, in new and more demanding decision problems, have indicated a great practical value of the method, but have also revealed the need to extend it in several directions, influenced by the following motivational arguments:

- DEX does not natively support structuring the attributes into *full hierarchies*, i.e., *directed acyclic graphs (DAG)*. Rather it manages the model structure as *trees*, enhanced with components called "links". Links in DEX are used in situations where a child attribute influences more than one parent attribute. Usage of multiple linked attributes decreases the model readability. We observed that usage of linked attributes was increasing through the years, especially in the decision models for usage in agronomy (Bohanec et al., 2009, 2008). We noted that DEX requires model structuring with full hierarchies, similar to the widely recognized method AHP (Saaty, 2008).

- Usage of numeric attributes is not possible in DEX, hence all input attributes must be discretized before usage in a DEX model. We observed a need to additionally support the numeric value scales in attributes, along with the qualitative value scales. With that, concepts naturally represented with numeric values (e.g. age of a person) can be represented in DEX in a natural way.

- With added support for numeric values in DEX, we also observed a need to facilitate aggregation of various combinations of value types (numeric and/or qualitative). For every combination of the input value types at least one aggregation function must exist that allows for aggregation of such values. Further, every combination of input values must be able to produce either a numeric or qualitative value. With this extension, DEX method would be able to dynamically translate numeric values into qualitative, and vice-versa, without any DM's manual intervention.

- DEX in its basic form handles crisp and interval values, however many examples of decision problems exist which have uncertain data and/or preferences provided. To

tackle decision problems with uncertain information, DEX needs to provide means to include such information in the models. Crisp and interval values (qualitative or numeric) need to be extended to include probabilistic distributions and fuzzy sets as model input values, and as outputs of aggregation functions. With inclusion of uncertainties there is a possibility for usage of advanced methods with DEX: model revisions with new unseen data (Žnidaršič, Bohanec, & Zupan, 2006a), and model hierarchy induction (Zupan, Bohanec, Demšar, & Bratko, 1999).

- In data storage and representation environments, data is not being presented as flat (in a single table), rather there are plenty connections between different sources of data that influence each other. One type of these connections are relational (*one-to-many*) connections, where multiple instances of objects influence the same parent object. This was observed particularly in the assessment of public e-portals (Leben et al., 2006), where each portal is influenced by multiple life events, and each life event is influenced by multiple e-services. Introduction of relational aspects into DEX would facilitate modeling relational connections as additional information for alternatives. Additional (relational) information can subsequently be modeled in a separate model, which is connected to the "main" model — providing the needed relational evaluations. With that, any previously manual aggregation of alternative's properties is eliminated.

On this basis, five substantial extensions (Trdin & Bohanec, 2012, 2014a) were identified. The aim is to extend DEX with additional features in order to address new decision making scenarios. The extensions should be formalized on top of DEX so that the basic formalization of the method remains intact. The extensions should be implemented in a computer library.

1.1 Aims and Hypothesis

The first and main aim of this dissertation is to extend the DEX methodology with features that are currently missing and are needed, so that the methodology would cope with increasing demands of decision makers and their decisions. The DEX method is formally extended into five directions:

- Native support for full hierarchies of attributes — to allow for criteria which influence more than one parent in the hierarchy.
- Numerical attributes are incorporated into the methodology in order to support decision problems which require combined usage of qualitative and quantitative modeling.
- Because of the extension of DEX methodology to numerical values, the current qualitative functions are insufficient to support all types of aggregation. With that in mind, we employ general aggregation functions which support decision makers to aggregate different combinations of numeric and qualitative inputs into a qualitative or numeric output.
- Probabilistic and fuzzy distributions are included in both, the level of alternatives and aggregation functions. This facilitates coping with uncertainty in models and data.
- Added support for relational modeling and aggregation of alternatives with relational models in order to support decision problems dealing with relational alternatives — that is, alternatives composed of several sub-components.

Additionally, the aim is to formally introduce the extensions in such a way that all extensions can be used together simultaneously in a single decision making scenario.

The second aim of this work is to develop a new implementation of the extended method, called *DEXx*. The implementation should provide the capabilities of method DEX together with the five developed extensions. The implementation should facilitate DMs in decision making with method DEX, providing framework for further development of the method. Furthermore, the implementation should be usable in different settings — standard shell programs, other developed programs, web sites and usage in web services.

The third aim of the dissertation is to evaluate the newly developed methodology and its implementation through real-life use-cases. Use-cases addressed in these scenarios have properties that, in order to be successfully solved, require the methodological extensions developed in this thesis. The four use-cases are:

- E-portals: using relational qualitative models for public e-portal evaluations (Leben & Bohanec, 2004; Leben et al., 2006).
- Reputational risk: usage of relational qualitative and quantitative models in reputational risk assessment of a bank. The use-case is derived from a European project titled FIRST (Project FIRST, 2013).
- Sustainability assessment of energy options (B. Kontić, Bohanec, Trdin, et al., 2014; B. Kontić et al., 2014b, 2016) using relational models with numeric and qualitative attributes to asses energy options in Slovenia until year 2050.
- Water flows: using modules for the evaluation of different water outflows from a field (drainage, infiltration, rapid infiltration, and saturation) in the domain of ecology and agronomy. The use-case is derived from an industrial project titled EVADIFF (Kuzmanovski, 2012; Kuzmanovski et al., 2015).

The main hypothesis of the thesis is that the implementation of the decision support method DEXx facilitates the development of decision models in complex situations, which were hard or even impossible to address previously with DEX. The implementation is able to cope with the ever increasing amounts and complexity of data, and handle this data in a timely fashion using the developed models.

We also expect that the extension of the DEX methodology will stimulate decision makers to think in a different manner, and with that construct new decision making models, which were not possible or considered too complex until now. Even more, we expect that the implementation and extensions of the methodology will present a new understanding of the methodology and its benefits.

1.2 Research Methodology

In order to prove the hypothesis, we used the following methodology.

First, we reviewed the existing decision making methods, and outlined their features — their negative and positive sides. We continued investigating problems the decision experts had, while using the DEX methodology and which features the methodology lacks. We reviewed the missing features and outlined the added benefits and weaknesses these features would bring to DEX. We identified five features, which need to be added to DEX methodology.

Then the requirements for identified features were recognized and formal description for the extensions was developed, keeping in mind the original DEX formal framework.

Based on the formal description, an implementation was developed, supporting usage in applications. All important extensions were formalized, presented and justified.

The requirements for the implementation were to develop the existing DEX methodology in Java programming language. The implementation is backward compatible with models, developed by the existing DEXi software package (Bohanec, 2014, 2015b) — reference implementation of DEX method, and is transferable among many computer platforms. On top of the basic DEX methodology, all five proposed extensions were implemented. The implementation supports using all extensions simultaneously, in the same decision making problem.

Finally, we constructed four use-cases which could not be carried out before, using any other decision making methodology. Developed decision models relied on the implementation of the DEXx methodology. The use-cases show that development of complex models, which were previously hard or even impossible to implement, is now possible to implement in a native fashion, with inclusion of the developed extensions and with the implementation of DEXx.

In order to evaluate DEXx and justify the development of such extensions, we assessed each use-case with the following criteria:

- Implementation of the model is possible, using DEXx formalization and its implementation.
- Without at least one of the proposed extensions, the implementation of the decision model would not be possible.
- The results of evaluations are identical, compared with the previous evaluations (if available).
- Evaluation time of alternatives in the model is decreased, in comparison with previous evaluation time (if available).

1.3 Scientific Contributions

Scientific contributions of the thesis are as follows:

- Development and formalization of the DEX methodology and formalization of the DEX extensions. Specific extensions are: full attribute hierarchies, numeric attributes, general aggregation functions, probabilistic and fuzzy extension of the domain, and relational models. Furthermore, the most important scientific contribution in this thesis is the analysis and implementation of *all* five extensions together in a unified framework. The framework facilitates using all extensions in the same decision making problem.
- Design and implementation of DEXx in one library, with high potential of reusability and transferability between different computer platforms. The implementation provides an easier access to the developed library to facilitate usage in different applications.
- Construction, application and evaluation of four complex use-cases using the new implementation of the extended methodology, which previously were not implementable in any other decision making methodology's framework:
  - Implementation of a qualitative relational model with a custom aggregation function, for native aggregation of relational alternatives.

- Implementation of a qualitative and quantitative relational model with high data input processing capabilities.
- Implementation of a qualitative and quantitative relational model with fixed alternatives and additional capability for simulating the model through time.
- Implementation of a qualitative model with regular alternatives, in a decision support system with high data-processing capabilities.

1.4 Organization of the Thesis

The thesis covers the related work in Chapter 2. Formal description of method DEX is given in Chapter 3. Chapter 3 also introduces an illustrative example which is developed throughout the thesis, with the described DEX formalization. In Chapter 4, the five extensions are presented, discussed, and developed formally. The introduced extensions are presented in a context of previously discussed illustrative example in Chapter 5. Chapter 6 presents the overview of the implementation, implementational facts, and important methods included. Chapters 7, 8, 9 and 10 present the four selected use-cases, and Chapter 11 concludes the thesis.

Chapter 2

Related Work

Decision making is a process in which a decision maker (DM) needs to select an alternative among several possible alternatives, which best satisfies his/her goals (Bouyssou et al., 2006; Figueira et al., 2005; French, 1986). Decision analysis (Clemen & Reilly, 2001; Nagel, 1993; Skinner, 2009) is a discipline that provides a framework for analysing decision problems, which typically involves the development of a model for the evaluation and analysis of alternatives. A common class of models employs methods of multi-criteria decision analysis (MCDA), where alternatives are evaluated using multiple, possibly conflicting, criteria (Bouyssou et al., 2006; Figueira et al., 2005; Ishizaka & Nemery, 2013).

MCDA is the activity of the person who, through use of explicit but not necessarily completely formalized models, helps obtain elements of responses to the questions posed by a stakeholder in decision process (Figueira et al., 2005). MCDA methods help the DMs to model given decision problems with their knowledge and possibly recommend the best alternative.

MCDA methods usually present the introduced alternatives with one of the three typical paradigms for recommendation of alternatives (Figueira et al., 2005):

- *Choosing* is oriented towards selecting a small subset (small as possible) of best possible alternatives so that the best final alternative may be chosen. Alternatives may be compared directly, in order to minimize the size of the produced set of best alternatives.

- *Sorting* (or *classification*) tries to assign each alternative to one of the predefined categories. The categories should represent different possible treatments of alternatives assigned to a particular category in the sorting. In sorting, the categories are preferentially ordered, whereas in classification, the categories are not ordered.

- *Ranking* aids the DM to rank the alternatives in a *complete* or *partial* order. With this approach every pair of alternatives can be directly compared — one can be preferentially better than the other or they are incomparable.

Most MCDA methods use one of the typical approaches to arrive to final decisions. The approaches (Figueira et al., 2005, p. 300–301) differ on the data they use for making decisions and on the way they acquire and use DM's preferences:

- *Value system approach* uses DM's preferences and information about the problem to aggregate given preferences on criteria, and arrive to *value* or *utility* system. The objective of this approach is to construct an actual value system, which later facilitates the evaluation of the alternatives.

- *Outranking approach* uses DM's preferences and information about the problem to construct outranking relations between the *predefined* alternatives. Here, the objective is to identify the *pros* and *cons* of the alternatives, and obtain a ranking between them. Exploitation of constructed relations allows the DM to make a "good" decision.

- *Disaggregation-aggregation approach* is an iterative process where components of the problem and DM's judgement policy are analyzed to arrive to a value system. The goal of this approach is to improve DM's knowledge about the decision situation (through analysing the behavior and the cognitive style of the DM), through which a consistent decision can be achieved.

- *Rule preference model approach* was developed for the *Dominance-based Rough Set Approach (DRSA)*. The approach gives the possibility of inferring a rule preference based model (with statements **if** ... **then** ... **else**), from exemplary made decisions. Its benefit is that the model is expressed in natural language, omitting any complex analytical formulation.

- *Multiobjective optimization approach* provides alternative evaluations based on a given multiobjective mathematical programming formulation. Here, the goal is to arrive to the actual mathematical program which is used for evaluation of alternatives. The approach can use different multiobjective optimization techniques to optimize the given formulation.

We review related work for methods in numeric multi-criteria analysis (MCDA), qualitative MCDA, use of hierarchies in MCDA, use of uncertain information in MCDA, and relational support in MCDA. Further, we stress the importance of methods for this thesis, and supply their possible weaknesses.

2.1 Multi-Criteria Decision Modeling

Multi-criteria decision modeling (MCDM) is a sub-discipline of operations research, which considers multiple criteria in decision making. The purpose is to support DMs in decision making problems. Typically, there is no unique optimal solution among the possible alternatives, hence it is necessary to use the DM's preferences to identify the *best* alternative (Bouyssou et al., 2006; Figueira et al., 2005).

The multi-criteria decision making problem can be represented in the criterion space $Q \subseteq \mathbb{R}^n$, where n is the number of criteria. Among the possible alternatives $A \subseteq Q$, MCDM is interested in the alternative that performs "well" in all possible criteria. Typically, alternatives perform well in some criteria and perform poorly in others. Finding a trade-off between the two is the purpose of MCDM.

Given n monotone increasing criterion functions $f_i, 1 \leq i \leq n$, the maximization MCDM problem can be stated as:

$$a' = \text{argmax}_{a \in A} f(a) = (f_1(a), f_2(a), \ldots, f_n(a)), \tag{2.1}$$

finding an alternative a' which maximizes all of the criterion functions.

2.2 Numeric Multi-Criteria Decision Modeling

In numeric multi-criteria decision modeling, all criteria have a numeric scale, hence all alternative's values are represented by tuples of numeric values. Aggregation functions

are again defined on numeric values and produce numeric criterion values. Generally, a numeric MCDM model evaluating alternatives $a \in A$ is a function $f : Q \mapsto \mathbb{R}$:

$$f(a) = \sum_{i=1}^{n} \lambda_i u_i(a), \tag{2.2}$$

where $\lambda_i \in [0, 1]$ and $\sum_{i=1}^{n} \lambda_i = 1$. λ_i represents the *weight* (or *importance*) of the i-th criterion. u_i is a function representing the performance of alternative a on criterion i. Typically, the function maps $u_i : Q \mapsto [0, 1]$.

Note, when u_i simply maps to the i-th value of alternative a ($u_i(a) = a_i$), function f is a simple weighted linear model. The model can then be simplified to:

$$f(a) = \sum_{i=1}^{n} \lambda_i a_i. \tag{2.3}$$

Typical representative methods that use numerical variables for values and preferences are: Kepner-Tregoe, UTA (UTilitiés Additives) and the Analytic Hierarchy Process (AHP).

Kepner-Tregoe (Kepner Higgins & Tregoe, 1997) is a value system method that computes numeric evaluations of alternatives by computing a weighted product of alternative's values with corresponding weights of criteria. The weights are assigned to criteria by the DM. DM also defines the mapping of non-numeric values of alternatives to numeric values.

UTA (Jacquet-Lagrèze & Siskos, 1982, 2001) is a disaggregation-aggregation method aimed at constructing additive value functions from a given ranking on a reference set of alternatives. Method uses linear programming to construct these functions so that the obtained functions are as consistent as possible with the reference set. The constructed model is a weighted sum of these additive functions that map an alternative's value to a computed numeric value.

In AHP (Saaty, 2008; Saaty & Vargas, 2012; Saaty, 2005), the value model is developed as a decomposition of criteria, which is induced from the decision goal (root of the hierarchy) into smaller sub-problems. The terminal nodes of the hierarchy are criteria that can be directly measured on the alternative. In each of the non-terminal nodes, a series of pairwise priorities regarding criteria and alternative values are expressed by the DM. Usually these priorities are expressed with numbers from interval $[1, 9]$ — here, the number represents the equal importance (1) of two alternatives (or criteria), or largest importance of the first alternative (9) in comparison to the second. Values in-between represent some smaller degree of importance. These pairwise judgements of sub-criteria result in a global priority for a particular aggregated criterion. Pairwise judgements can be checked for consistency (e.g. if DM was consistent during expressing preferences). Finally, the alternatives are evaluated according to global priorities and alternative value priorities. AHP is one of the most popular MCDM techniques, however the basic AHP does not cope well with a large number of alternatives, because the number of pairwise comparison between alternatives grows quadratically in terms of the number of alternatives.

Numeric attributes are relevant to this thesis in order to introduce numerical attributes in DEX. The main difficulty for including numeric attributes in DEX is the aggregation of different combinations of numeric and qualitative values. We overcome this difficulty by providing the DM new means for aggregation of different combinations of numeric and qualitative values. All combinations may be aggregated into either numeric or qualitative values. This approach also includes the needed discretization of numeric values.

2.3 Qualitative Multi-Criteria Modeling

Qualitative multi-criteria methods are characterized by using qualitative variables whose value scales contain a finite predefined set of qualitative (or symbolic) values (for example, "low", "medium", "high"). Methods in qualitative MCDA differ in the way they model decision problems and how they acquire DM's preferential information.

Values in numeric MCDM are expressed with numbers (qualitative values must be mapped to numeric values), consequently the scales are usually preferentially ordered. In contrast, values in qualitative MCDM methods allow for expressing non-determined (vague or fuzzy) information — consequently, scales can be ordered or unordered. Further, methods in numeric MCDM aggregate values using mathematical functions, such as the weighted sum. Methods in qualitative MCDM frequently interpret decision tables and aggregation functions as collections of *if-then* rules.

Qualitative MCDM methods are particularly important for providing descriptive or non-numeric information, while still providing useful information for the modeling process (for example, *attractiveness* of a car is not directly measurable, while it has meaning). Qualitative values in MCDM provide means to express *fuzzy* values — values that numerically have meaning, however are too specific to explicitly consider (for example, consumption of a car being 15l/100km is naturally expressible as *high*). DMs tend to think in qualitative means more than in numeric values (Figueira et al., 2005, p. 508).

There are two groups of qualitative MCDM methods which differ in the way knowledge is acquired from the DM while building a decision model (Bohanec, Rajkovič, et al., 2013; Boose et al., 1993): (1) methods based on an interactive questioning procedure for obtaining the DM's preference, and (2) methods that acquire the DM's preferences directly.

Representative methods of the interactive questioning procedure are MACBETH, ZAPROS and ORCLASS.

MACBETH (Measuring Attractiveness by Categorical Based Evaluation Technique) (Bana e Costa & Vansnick, 1999) uses attractiveness and differential judgments between attributes in order to build preferential relations between alternatives.

ZAPROS and ORCLASS are methods belonging to VDA (Verbal Decision Analysis) (Larichev & Moshkovich, 1994, 1997; Moshkovich & Mechitov, 2013).

ZAPROS (a Russian abbreviation for Closed Procedures near Reference Situations) (Larichev, 2001; Larichev & Moshkovich, 1995) provides outranking relationships among alternatives through a verbal decision making approach. It is designed to support a large number of alternatives, but it does not cope well with large amounts of criteria.

ORCLASS (ORdinal CLASSification) (Gomes, Moshkovich, & Torres, 2010) assigns alternatives to predefined, ordered classification categories, where alternatives are placed into classes using a set of criteria.

Typical representatives of the second group are the methods DRSA, Doctus and DEX.

DRSA (Dominance-based Rough Set Approach) (Greco, Matarazzo, & Słowiński, 2001, 2002) uses rough sets theory with the goal of solving alternative classification and sorting problems represented by decision tables, using the principle of dominance. DRSA has a strong mathematical foundation (Greco et al., 2001) and has evolved in many directions — for example, considering imprecise evaluations and assignments (Dembczyński, Greco, & Słowiński, 2009) and dealing with decisions under uncertainty and time preference (Greco, Matarazzo, & Słowiński, 2010).

Doctus (Baracskai & Dörfler, 2003) is a Knowledge-Based Expert System Shell used for evaluation of alternatives and supporting three types of alternative evaluation: Rule-Based Reasoning (RBR), Case-Based Reasoning (CBR) and Case-Based Rule Reasoning (CBRR).

The third method from this group is DEX, which is addressed in this work. DEX (Decision EXpert) is a qualitative multi-attribute modeling method which integrates multi-criteria decision modeling with rule-based expert systems (Bohanec, Rajkovič, et al., 2013). A DEX model consists of hierarchically structured qualitative attributes whose values are words rather than numbers. The aggregation is defined by decision rules. DEX has been widely used in practice to support complex decision processes in various fields of study. DEX copes well with a large number of alternatives, because the constructed model is not dependent on the number of alternatives.

Models similar to DEX were proposed by Arthur Lee Samuel for use in machine learning. Model learning proposed in (Samuel, 1967) proved to be very successful at the task of playing a game of checkers. Later, in (Zupan, Bratko, Bohanec, & Demšar, 2001) method HINT was developed, which is able to learn DEX models from decision examples. Further, Donald Michie proposed a machine learning structuring approach to decompose the considered problem to sub-problems (Michie, 1995), similar to approach in DEX.

DEX is described in more detail in Chapter 3.

Qualitative attributes are the most basic building block in DEX method. Since the thesis is concerned with extending DEX, qualitative values together with rule-based aggregation functions are of great importance to this work, and are formalized further in Chapter 3 and extended in Chapter 4.

2.4 Hierarchies in Multi-Criteria Decision Modeling

Most notable usages of hierarchies are displayed in AHP (Saaty, 2008; Saaty & Vargas, 2012; Saaty, 2005) and DEX (Bohanec, Rajkovič, et al., 2013). In both cases the decision problem is decomposed from more complex concepts to smaller, more manageable concepts. Each concept devised in that way correlates to some concrete concept on the alternative. Concepts that are joined into a parent (*aggregated*) concept typically form a logical concept in terms of the decision problem. For example, while choosing a family car, *number of doors* and *luggage space* logically form the *commodity* concept.

In principle, a DEX model can be a hierarchy. So far, hierarchies were only indirectly supported in DEX (Bohanec, 2015a; Bohanec & Rajkovič, 1990; Bohanec, Rajkovič, et al., 2013) and DEXi software (Bohanec, 2015b, 2014), using the concepts called "chaining" and "linking" of nodes. Full hierarchies were supported in DEX through proDEX software package (Žnidaršič et al., 2006b) and were used for modeling the impacts of growing genetically modified maize (Žnidaršič et al., 2008).

The idea of hierarchies in this thesis is borrowed from AHP. We wish to eliminate "links" from the method and natively allow structuring of decision criteria into a hierarchy.

2.5 Uncertainties and Fuzziness in Multi-Criteria Decision Modeling

The extension of probabilistic distributions incorporates ideas from probabilistic inference methods (Durbach & Stewart, 2012; Shachter & Peot, 1992; J. B. Yang, Wang, Xu, & Chin, 2006). Actually, these ideas have already been addressed in DEX to some extent. An early version of the method, called DECMAK, used probabilistic and fuzzy distributions for the representation of alternatives' values (Rajkovič et al., 1987). Probabilistic distributions were implemented in HINT (Hierarchy INduction Tool) (Zupan et al., 1999) for usage in function decomposition in machine learning (Zupan et al., 2001). In order to facilitate the revision of the models (Žnidaršič et al., 2006a), probabilistic distributions were introduced

to aggregation functions and implemented in the system called proDEX (Žnidaršič et al., 2006b).

Some steps towards a decision making method with the probabilistic extension were presented in (Trdin & Bohanec, 2012, 2013, 2014a) and were applied in a real-life decision making scenario in (B. Kontić, Bohanec, Trdin, et al., 2014; B. Kontić et al., 2016). Similar approaches were explored by other authors (Bergez, 2013; Holt et al., 2013; Omero, D'Ambrosio, Pesenti, & Ukovich, 2005).

This thesis extends DEX in order to allow representation of values as value distributions — most notably, probabilistic distributions and fuzzy distributions. Considering that this thesis also introduces numeric attributes, numeric values distributions are also needed. Inclusion of value distributions in DEX poses a problem during aggregation: while aggregating more input values as value distributions, their respective properties must be propagated to the evaluation result (for example, probabilities of input probabilistic distributions must influence the evaluation's probabilistic distribution). Even more, combinations of qualitative and numeric value distributions in same aggregation function pose an even more difficult problem, which we overcome in Chapter 4.

2.6 Relational Multi-Criteria Modeling

Data in complex environments (such as companies) are frequently stored in relational models, more commonly referred to as *relational database*. The model is based on *first-order predicate logic*, where data is represented as *tuples*, and their relationships are modeled by *relations*. There are different combinations of multiplicities of the relations: *one-to-one*, *one-to-many* and *many-to-many*. Here, one-to-one corresponds to a relation which maps *one* tuple from the first table to *one* tuple from the second table. One-to-many means that *one* tuple from the first table corresponds to *many* tuples from the second table. Similarly, many-to-many refers to *many* tuples from the first table correspond to *many* tuples from the second table. In this thesis we are interested in the relation *one-to-many*, where we want to assess the *one* alternative on the basis of the corresponding *many* other alternatives. For example, while evaluating a *company*, all of the corresponding *departments* need to be assessed before we can evaluate the company in whole.

Several disciplines of machine learning explicitly consider the development of relational models — for example, inductive logic programming (Lavrač & Džeroski, 1994). Relational data are also considered in quantitative multi-criteria decision making methods, but rarely in an explicit way. There, it also rarely causes difficulties because it naturally involves common aggregation operators based on summation and averaging while at the same time handling numeric values. Such operators are useless in a qualitative setting and require special approaches.

Relational aggregation in MCDM is usually considered for portfolio management — in stocks or company project selection. Selection of common stock portfolio was modeled with *outranking* approaches, and then combined with *non-linear optimization model* in (Xidonas, Askounis, & Psarras, 2009). Authors Yu, Wang, Wen, and Lai (2012) considered project portfolio selection with *non-linear integer programming*, solving it with *genetic algorithms*. Authors Arasteh, Aliahmadi, and Omran (2014) also considered company project portfolio selection, tackling the problem with *Goal Programming*. They employ fuzzy conditions for portfolio prices and criteria. Hurson and Ricci-Xella (1998) employ *Arbitrage Pricing Theory* for estimating the project's expected return in portfolio management process. Further, they use *ELECTRE TRI* and *MINORA interactive system* to select an attractive portfolio. In (Jiang, Zhang, & Sutherland, 2011), the authors developed a model for selection of a remanufacturing technology portfolios, promoting environmen-

tal sustainability. They employ *AHP* for pairwise comparison of portfolios. From this literature we observe that there is no unified way to tackle relational data in the form of projects (or stocks). Each study employs different evaluation of projects (stocks) and different *selection* method to maximize the return.

We also encounter relational alternatives in group decision making, where all the decision makers have different preferences on the same matter — the matter can be treated as some part of the sub-hierarchy. The top aggregation function, where the combination of all sub-model evaluations are combined, is the most important — the aggregation is not constrained just to calculating simple functions, but it can have a more complex structure. A similar technique was already implemented in DEX software (Bohanec, 2015a; Bohanec & Rajkovič, 1990; Bohanec, Rajkovič, et al., 2013) as "groups", but in a limited fashion.

Support for relational models is one of the identified extensions in this thesis, and requires special treatment, since qualitative relational aggregation is not as developed as relational aggregation of numeric values. Inclusion of relational concepts requires understanding the needs for representation of relational alternatives, and determining types of aggregation of qualitative evaluations.

Chapter 3

Qualitative Multi-Criteria Method DEX

DEX (Bohanec, 2015a; Bohanec & Rajkovič, 1990, 1999; Bohanec, Rajkovič, et al., 2013; Bohanec & Trdin, 2014; Trdin & Bohanec, 2012, 2014a) is a qualitative multi-criteria decision modeling method. It enables the decision maker to structure the decision problem into smaller and easily measurable concepts (attributes). The attributes aggregated into a common aggregated attribute form a logical concept. Structure of attributes forms a hierarchy. DEX models have one or more *root* attribute(s) — attributes which do not have any parent attributes. Attributes which do not have any child attributes are called *input attributes*; all other attributes are *aggregated attributes*.

Attributes in DEX use qualitative scales which consist of a finite set of symbolic attribute values, such as "bad" "medium" and "good", rather than numeric values. These values are usually, but not necessarily, preferentially ordered. Consequently, a DEX model generally consists of attributes, some of which are preferentially ordered and can be thus referred to as criteria.

To aggregate qualitative values, DEX uses decision tables, which can be interpreted as collections of *if-then* rules. Every aggregated attribute has an aggregation function assigned. To define an aggregation function, the DM defines an output value for each combination of input attribute's values.

Alternatives in DEX are defined by qualitative values, which are assigned to the input attributes of the model. DEX evaluates alternatives in a bottom-up way, progressively computing the aggregation functions, for which their respective children attributes already have a value assigned. The final evaluation of an alternative are the values computed at the roots of the model.

In this chapter we present the main entities in DEX method — models, attributes, aggregation functions, alternatives and evaluation of alternatives. These entities are presented formally and on an *illustrative example* through the chapter. Furthermore, we present the software implementations that support modeling with method DEX.

3.1 DEX Model

A DEX *model* has a form of a *hierarchy*, which represents the decomposition of the decision problem and relations between attributes: higher-level attributes depend on lower-level ones. The terminal nodes of the hierarchy (nodes without any children in the hierarchy) are called *input* or *basic attributes*, whereas all other attributes are called *aggregated attributes*. Additionally, attributes without any parents are called *roots*. A typical model has only one root, which represents the primary outcome of the evaluation of alternatives. There are

no conceptual limitations, however, to having multiple roots — for example, to represent evaluations from different viewpoints.

Formally, a DEX model M is a four-tuple $M = (X, D, S, F)$, where X is the set of attributes, S is the descendant function that determines the hierarchical structure of M, D is the set of value scales (domains) of attributes in X, and F is the set of aggregation functions.

The set X consists of n attributes:

$$X = \{x_1, x_2, \ldots, x_n\}. \tag{3.1}$$

In practice, attributes are usually given a name, which uniquely identifies the attribute — for instance "price", "quality", "location", etc. In practice, we often denote an attribute by its name (e.g. *location*) and use a named subscript to denote related components (e.g. D_{location}).

Each attribute $x_i \in X$ has a corresponding *value scale* $D_i \in D$, which is an ordered set of symbolic (qualitative) values:

$$D_i = \{w_{i,1}, w_{i,2}, \ldots, w_{i,m_i}\}. \tag{3.2}$$

Here, m_i is the number of values in the scale of attribute x_i. In practice, attribute values are also represented by words, such as "low", "good", "acceptable" and "yes".

With respect to the DM's preferences, scales can be either *preferentially ordered* or *preferentially unordered*. If the scale is ordered, then the DM defines the preferential operator $\preceq$, which is a total order. In this case, for each $j \leq k, j, k \in [1, m_i]$, we write $w_{i,j} \preceq w_{i,k}$ and interpret that $w_{i,k}$ is preferentially at least as good as $w_{i,j}$.

Furthermore, if additionally $w_{i,j}$ is preferentially at least as good as $w_{i,k}$ ($w_{i,k} \preceq w_{i,j}$), we say $w_{i,j}$ and $w_{i,k}$ are equivalent, denoted by $w_{i,j} \approx w_{i,k}$.

For ordered scales, we use the notation:

$$D_i = (w_{i,1}, w_{i,2}, \ldots, w_{i,m_i}). \tag{3.3}$$

According to the usual convention (Greco et al., 2001), attributes with preferentially ordered scales are called *criteria*.

Attributes are structured hierarchically: each attribute x_i may have some descendants (children) and/or predecessors (parents) in the model. This relationship is described by the function $S : X \mapsto 2^X$, which maps each attribute $x \in X$ to a set of its descendants $S(x)$:

$$S(x) = \{x_i, x_j, \ldots, x_k\}, x_i, x_j, \ldots, x_k \in X. \tag{3.4}$$

The relations induced by S must represent a hierarchy — that is, a connected and *directed acyclic graph (DAG)*.

Currently, in DEX (Bohanec, 2015a; Bohanec & Rajkovič, 1990; Bohanec, Rajkovič, et al., 2013) and its implementation DEXi (Bohanec, 2015b, 2014), hierarchies are handled with the so-called "links". The model is developed as a tree, but DEXi creates an implicit connection between attributes with the same name and scale. Internally the model structure is still considered to be a tree. Because of this, only one of the *linked* attributes may have descendant attributes. The main difference is evident during the evaluation of alternatives — when evaluating a linked attribute, all other linked attributes receive the same computed value. This approach is feasible when there are just a few linked attributes present in the model. It can be difficult for the DM to interpret the model with more linked attributes due to implicit connections.

In most practical cases, the graph (induced by S) represents a tree, in which all attributes, except a single root attribute, have exactly one parent. In general, however, a hierarchy may have several roots and may contain attributes that influence more than one parent attribute. If S does not form a hierarchy, then the graph contains one or more *cycles*, and the evaluation of alternatives is not possible.

Now that the function S is defined, we can induce three important sets of attributes: $modelInputs_M$, $modelOutputs_M$ and $aggAttributes_M$. Alongside, two important Cartesian products of attribute scales are induced: $inputSpace_M$ and $outputSpace_M$.

The model input attributes, $modelInputs_M$, constitute a set of all attributes that do not have any children in the hierarchy.

$$modelInputs_M = \{x_i \in X | S(x_i) = \emptyset\}. \tag{3.5}$$

Additionally, the model input space $inputSpace_M$ is defined as the Cartesian product of attributes' scales in $modelInputs_M$:

$$inputSpace_M = D_j \times D_l \times \cdots \times D_k, x_j, x_l, \ldots, x_k \in modelInputs_M. \tag{3.6}$$

Model output attributes and model output space are defined similarly. $modelOutputs_M$ is a set of all attributes in model M that are not children to any attribute in the hierarchy, whereas the $outputSpace_M$ is the Cartesian product of these attributes' scales:

$$modelOutputs_M = \{x_i \in X | x_i \notin \bigcup_{x \in X} S(x)\}, \tag{3.7}$$

$$outputSpace_M = D_j \times D_k \times \cdots \times D_l, x_j, x_l, \ldots, x_k \in modelOutputs_M. \tag{3.8}$$

Another useful set is the set of all aggregated attributes, $aggAttributes_M$. These attributes are the ones whose values have to be computed when evaluating alternatives. Structurally, they appear as internal nodes in the hierarchy:

$$aggAttributes_M = \{x_i \in X | x_i \notin modelInputs_M\}. \tag{3.9}$$

3.1.1 Illustrative Example

We will follow the same illustrative example through this chapter and show the usage of presented extensions later in Chapter 5. The example addresses the decision problem of choosing an *apartment* to buy. At the first stage, the model is developed using the DEX method without any extensions.

The model has one *root* attribute, *apartment*, and three children: *price*, *location* and *layout*. These three attributes are split into finer detail, which can be seen in Table 3.1. In this way, *price* depends on *buying price* and the *price of utilities*. *Location* is considered an input attribute, and assesses the location of the apartment. The third sub-tree assesses the *layout* of the apartment, considering its *interior* and *exterior* — these are both input attributes.

All of the attributes have qualitative values assigned to their scales, which is also shown in Table 3.1, on the right to the attributes' names. Notably, the model has a simple tree structure, in which none of the attributes influence more than one parent attribute. In total, there are 8 attributes: 5 input and 3 aggregated, including one root attribute.

Formally, the model has the following attributes:

$$X = \{\text{apartment}, \text{price}, \text{buying price}, \text{utilities}, \text{location}, \text{layout}, \text{interior}, \text{exterior}\}. \tag{3.10}$$

Table 3.1: The model for qualitative assessment of apartments. On the left side, the structure of the attributes is presented, and on the right side, their corresponding scales are displayed. The scales are ordered from worst values (displayed in red) to best values (displayed in green).

Attribute	Scale
apartment	very bad, bad, ok, good, very good
price	high, medium, low
buying price	high, medium, low
utilities	high, medium, low
location	bad, ok, good
layout	bad, ok, good
interior	bad, ok, good
exterior	bad, ok, good

Further, the preferentially ordered scales presented in the model are the following:

$$\begin{aligned} D_{\text{apartment}} &= (\text{very bad}, \text{bad}, \text{ok}, \text{good}, \text{very good}), \\ D_{\text{price}} &= (\text{high}, \text{medium}, \text{low}), \\ D_{\text{buying price}} &= (\text{high}, \text{medium}, \text{low}), \\ D_{\text{utilities}} &= (\text{high}, \text{medium}, \text{low}), \\ D_{\text{location}} &= (\text{bad}, \text{ok}, \text{good}), \\ D_{\text{layout}} &= (\text{bad}, \text{ok}, \text{good}), \\ D_{\text{interior}} &= (\text{bad}, \text{ok}, \text{good}), \\ D_{\text{exterior}} &= (\text{bad}, \text{ok}, \text{good}). \end{aligned} \tag{3.11}$$

Table 3.1 also shows the descendants of the attributes, and hence we can define the S function:

$$\begin{aligned} S(\text{apartment}) &= \{\text{price}, \text{location}, \text{layout}\}, \\ S(\text{price}) &= \{\text{buying price}, \text{utilities}\}, \\ S(\text{buying price}) &= \{\}, \\ S(\text{utilities}) &= \{\}, \\ S(\text{location}) &= \{\}, \\ S(\text{layout}) &= \{\text{interior}, \text{exterior}\}, \\ S(\text{interior}) &= \{\}, \\ S(\text{exterior}) &= \{\}. \end{aligned} \tag{3.12}$$

With the descendants defined, we can define the model inputs and input space, *model Inputs*$_{\text{apartment}}$, *inputSpace*$_{\text{apartment}}$, respectively:

$$modelInputs_{\text{apartment}} = \{\text{buying price}, \text{utilities}, \text{location}, \text{interior}, \text{exterior}\}, \tag{3.13}$$

$$\begin{aligned} inputSpace_{\text{apartment}} = (\text{high}, \text{medium}, \text{low}) \times (\text{high}, \text{medium}, \text{low}) \times \\ \times (\text{bad}, \text{ok}, \text{good}) \times (\text{bad}, \text{ok}, \text{good}) \times (\text{bad}, \text{ok}, \text{good}). \end{aligned} \tag{3.14}$$

Similarly we can define $modelOutputs_{\text{apartment}}$ and $outputSpace_{\text{apartment}}$:

$$modelOutputs_{\text{apartment}} = \{\text{apartment}\}, \tag{3.15}$$

$$outputSpace_{\text{apartment}} = (\text{very bad}, \text{bad}, \text{ok}, \text{good}, \text{very good}). \tag{3.16}$$

With this we finished the structuring of attributes in the apartment model. We continue the development of the illustrative example with defining of aggregation functions in Section 3.2.3.

3.2 Aggregation Functions

The aggregation of values in the DEX model is facilitated by *decision rules*. In order to compute an aggregated attribute's value from the values of its children in the hierarchy, each aggregated attribute has an associated *total aggregation function*. The function is defined by a *decision table*, which can also be interpreted as a set of *decision rules*. The total aggregation function needs to specify an *output value* for *every combination* of children's values. Typically, such tables are prepared by the DM according to his/her preferences.

Each aggregated attribute $x_i \in aggAttributes_M$ needs an aggregation function defined so that the decision alternatives can be evaluated. We denote $f_i \in F$ as the aggregation function corresponding to attribute x_i. Each aggregation function f_i is a total function that maps all scale value combinations of x_i's inputs to an interval value in scale D_i:

$$f_i : D_j \times D_k \times \cdots \times D_l \mapsto I(D_i), \{x_j, x_k, \ldots, x_l\} = S(x_i). \tag{3.17}$$

Here, $I(D_i)$ represents the space of all possible intervals $[v_l, v_h] = \{v_j | v_l \preceq v_j \preceq v_h \wedge v_l, v_j, v_h \in D_i\}$. Intervals are used, rather than crisp qualitative values, for two reasons:

- The expressive power of the functions is greater with intervals than with crisp qualitative values. With intervals, an aggregation function can produce an evaluation between two edge *values* from domain D_i.
- Allowing intervals as outputs of aggregation functions in DEX, even partly-defined functions can be used for aggregation. For example, if a function is not defined for some input values, the lower limit and the higher limit of the evaluation can be deduced from other defined output values, according to the principle of dominance (see Section 3.2.2). For more information see (Bohanec, 2014).

Usually such aggregation functions are represented as tables which assign an output value to each possible input value combination. Such tables can also be interpreted as if-then rules. Examples of such tables are given in Tables 3.2, 3.3, and 3.4, in the context of the illustrative example.

Functions in DEX are defined on discrete points in Cartesian space of attribute's input values — representing all possible combinations of input values. For that matter, we can define the *completeness* of a function $f_i \in F$. completeness$_i \in [0, 1]$ measures "how well defined" is some function f_i — for how many input combinations the function produces a crisp value. Here, 0 means that function is empty, and 1 means that function produces a crisp value for every combination of input values.

To define the completeness$_i$, we must firstly define the completeness of one defined rule. Suppose we are given a function $f_i : D_j \times D_k \times \cdots \times D_l \mapsto I(D_i)$, then $d_i : D_j \times D_k \times \cdots \times D_l \mapsto [0, 1]$ represents the completeness of one rule:

$$d_i(v_j, v_k, \ldots, v_l) = \frac{|D_i| - |f_i(v_j, v_k, \ldots, v_l)|}{|D_i| - 1}, \tag{3.18}$$

here $|D_i|$ represents number of qualitative values in scale of x_i and $|f_i(v_j, v_k, \ldots, v_l)|$ represents the number of values in the returned interval value.

With that, we can define the completeness$_i$ of a function, for a set of defined rules $R \subseteq D_j \times D_k \times \cdots \times D_l$:

$$\text{completeness}_i = \frac{1}{|R|} \sum_{(v_j, v_k, \ldots, v_l) \in R} d_i(v_j, v_k, \ldots, v_l). \tag{3.19}$$

3.2.1 Generalized Functions

For the purpose of using functions f_i in the evaluation of alternatives, especially in the DEXx method described later, we also define functions F_i, which are *generalizations* of f_i and are defined on sets of values rather than on crisp values. The F_i function definition corresponding to the associated function f_i is:

$$F_i : 2^{D_j} \times 2^{D_k} \times \cdots \times 2^{D_l} \mapsto 2^{D_i}. \tag{3.20}$$

Function F_i is a generalization of f_i, and both are expected to produce the same results on the same singleton input values. Therefore, they are constrained as follows:

$$\begin{gathered} \forall (v_j, v_k, \ldots, v_l) \in D_j \times D_k \cdots \times D_l : W_j = \{v_j\}, W_k = \{v_k\}, \ldots, W_l = \{v_l\} : \\ F_i(W_j, W_k, \ldots, W_l) = \{f_i(v_j, v_k, \ldots, v_l)\}. \end{gathered} \tag{3.21}$$

Given f_i, the value for function F_i is computed as:

$$F_i(W_j, W_k, \ldots, W_l) = \bigcup_{(v_j, v_k, \ldots, v_l) \in W_j \times W_k \times \cdots \times W_l} \{f_i(v_j, v_k, \ldots, v_l)\}. \tag{3.22}$$

Here, all possible value combinations from $(v_j, v_k, \ldots, v_l) \in W_j, W_k, \ldots, W_l$ are used to compute the function value of f_i. Then, the union of all evaluations is created. From this definition we observe that the constraint from Equation (3.21) is satisfied.

3.2.2 Monotonic Functions

Decision modeling methodologies must somehow provide a guarantee for consistency in terms of dominance, e.g. the best selected alternative must not be preferentially worse than any other considered alternative. In DEX we can confirm consistency if all the aggregation functions are monotonic.

To define monotonic functions, we first have to define dominance relation on tuples.

Suppose we are considering a set of scales, $D_1, D_2, \ldots, D_k$, where each D_i is an ordered scale, e.g. $D_i = (w_{i,1}, w_{i,2}, \ldots, w_{i,k_i})$. Then a tuple of values $\overrightarrow{v_p}$ dominates a tuple of values $\overrightarrow{v_r}$, formally $\overrightarrow{v_r} \preceq \overrightarrow{v_p}$, when:

$$\begin{gathered} \overrightarrow{v_p}, \overrightarrow{v_r} \in D_1 \times D_2 \times \cdots \times D_k, \\ \overrightarrow{v_r} \preceq \overrightarrow{v_p} \iff \overrightarrow{v_{r,1}} \preceq \overrightarrow{v_{p,1}} \wedge \overrightarrow{v_{r,2}} \preceq \overrightarrow{v_{p,2}} \wedge \cdots \wedge \overrightarrow{v_{r,k}} \preceq \overrightarrow{v_{p,k}}. \end{gathered} \tag{3.23}$$

Here $\overrightarrow{v_{p,i}}$ and $\overrightarrow{v_{r,i}}$ denote the i-th value of the corresponding tuple $\overrightarrow{v_p}$ and $\overrightarrow{v_r}$. Note that the relation $\overrightarrow{v_i} \preceq \overrightarrow{v_i}$ always holds, where $\overrightarrow{v_i} \in D_1 \times D_2 \times \cdots \times D_k$.

Suppose, we have a function $f : D_j \times D_k \times \cdots \times D_l \mapsto D_i$, where all $D_i, D_j, D_k, \ldots, D_l$ are increasingly ordered (see Equation (3.3)). Then the function f_i is said to be monotone increasing if:

$$\begin{gathered} \forall \overrightarrow{v_p}, \overrightarrow{v_r} \in D_j \times D_k \times \cdots \times D_l, \\ \overrightarrow{v_r} \preceq \overrightarrow{v_p} \implies f_i(\overrightarrow{v_{r,j}}, \overrightarrow{v_{r,k}}, \ldots, \overrightarrow{v_{r,l}}) \preceq f_i(\overrightarrow{v_{p,j}}, \overrightarrow{v_{p,k}}, \ldots, \overrightarrow{v_{p,l}}). \end{gathered} \tag{3.24}$$

The rationale behind this statement is that if any argument in the function preferentially increases in value, then the functional value should not decrease. Because the outputs of function $v_i = f_i(x_j, x_k, \ldots, x_l)$ are always in the scale D_i of the corresponding attribute x_i, different values of v_i can again be compared with the preferential operator defined in Section 3.1.

Similarly, the function g_i is said to be monotone decreasing if:

$$\begin{gathered} \forall \overrightarrow{v_p}, \overrightarrow{v_r} \in D_j \times D_k \times \cdots \times D_l, \\ \overrightarrow{v_r} \preceq \overrightarrow{v_p} \implies g_i(\overrightarrow{v_{r,j}}, \overrightarrow{v_{r,k}}, \ldots, \overrightarrow{v_{r,l}}) \succeq g_i(\overrightarrow{v_{p,j}}, \overrightarrow{v_{p,k}}, \ldots, \overrightarrow{v_{p,l}}). \end{gathered} \tag{3.25}$$

The rationale behind monotone decreasing is that if any argument in the function preferentially increases in value, then the functional value should not increase.

Note that a constant function is by definition always monotone decreasing and monotone increasing.

To provide the consistency of the model we must guarantee that subsequent evaluations of monotone increasing functions in hierarchy will again result in consistent final evaluation. Suppose two monotone increasing functions are given: $f_j : D_l \times D_k \times \cdots \times D_o \mapsto I(D_j)$ and $f_i : D_j \mapsto I(D_i)$. The composition of two functions f_j and f_i is then $f_j \circ f_i : D_l \times D_k \times \cdots \times D_o \mapsto I(D_i)$, and is computed as $f_i(f_j(v_l, v_k, \ldots, v_o))$, for some $(v_l, v_k, \ldots, v_o) \in D_l \times D_k \times \cdots \times D_o$.

From the monotone increasing function definition stated in Equation (3.24) we observe that composition $f_i \circ f_j$ of two monotone increasing functions f_i and f_j is also a monotone increasing function. Note that the composition of functions can be generalized when f_i has more than one input — in this case, the monotone increasing property of the composition is preserved.

3.2.3 Illustrative Example

We continue with the development of the apartment illustrative example with defining the aggregation functions. With this step, the modeling procedure will be finished. We need to define an aggregation function for the three aggregated attributes: *apartment*, price and *layout*. The aggregation functions are described by sets of decision rules, represented with tables. Here note that the following presented tables map their values into crisp qualitative values, rather than intervals. This is done by convention that when the lower and the upper limit of the interval is the same *value*, the *value* is written as crisp.

The aggregation function for *apartment* aggregated attribute is presented in Table 3.2. The aggregation function aggregates values from *price*, *location* and *layout* into values from *apartment*. We observe that the function has a value defined for all possible input value combinations, therefore it is complete ($\text{completeness}_{\text{apartment}} = 1$) and total.

The Table 3.2 contains a set of decision rules, which can be interpreted as *if-then* rules. For example:

$$\begin{aligned} f_{\text{apartment}}(\text{high}, \text{ok}, \text{ok}) &= \text{very bad}, \\ f_{\text{apartment}}(\text{low}, \text{good}, \text{bad}) &= \text{good}. \end{aligned} \tag{3.26}$$

Table 3.2: The aggregation function for apartment evaluation from price, location and layout. Every combination of input values (on the left) results in an output value (right-most column).

price	*location*	*layout*	*apartment*
high	bad	bad	very bad
high	bad	ok	very bad
high	bad	good	very bad
high	ok	bad	very bad
high	ok	ok	very bad
high	ok	good	very bad
high	good	bad	very bad
high	good	ok	very bad
high	good	good	bad
medium	bad	bad	very bad
medium	bad	ok	very bad
medium	bad	good	very bad
medium	ok	bad	bad
medium	ok	ok	bad
medium	ok	good	bad
medium	good	bad	bad
medium	good	ok	ok
medium	good	good	ok
low	bad	bad	bad
low	bad	ok	bad
low	bad	good	ok
low	ok	bad	good
low	ok	ok	good
low	ok	good	good
low	good	bad	good
low	good	ok	very good
low	good	good	very good

These two can be interpreted as: **if** *price* is *high* **and** *location* is *ok* **and** *layout* is *ok* **then** *apartment* is *very bad*, and **if** *price* is *low* **and** *location* is *good* **and** *layout* is *bad* **then** *apartment* is *good*.

The aggregation function for aggregated attribute *price* is presented in Table 3.3. The aggregation function aggregates values from *buying price* and *utilities* into values from *price*. We see the function has a value defined for all possible input value combinations, hence it is complete ($\text{completeness}_{\text{price}} = 1$) and total.

Table 3.3: The aggregation function for price evaluation from buying price and utilities. Every combination of input values (on the left) results in an output value (right-most column).

buying price	*utilities*	*price*
high	high	high
high	medium	high
high	low	medium
medium	high	high
medium	medium	medium
medium	low	medium
low	high	medium
low	medium	low
low	low	low

The last aggregation function in the illustrative example is for the aggregated attribute *layout*, and is presented in Table 3.4. The aggregation function aggregates values from *interior* and *exterior* into values from *layout*. The function is again total and complete

$(\text{completeness}_{\text{layout}} = 1)$.

Table 3.4: The aggregation function for layout evaluation from interior and exterior. Every combination of input values (on the left) results in an output value (right-most column).

interior	*exterior*	*layout*
bad	bad	bad
bad	ok	bad
bad	good	ok
ok	bad	bad
ok	ok	ok
ok	good	ok
good	bad	ok
good	ok	good
good	good	good

In this example we can confirm that all three aggregation functions are monotone increasing, hence we can guarantee consistency of evaluations.

3.3 Alternatives and Evaluation of Alternatives

Alternatives in DEX are represented with qualitative values from scales of input attributes. Formally, an alternative a_i is a tuple from the Cartesian space $inputSpace_M$, for a decision model M, $a_i \in inputSpace_M$.

With A_M we denote the set of all decision alternatives defined for the decision problem:

$$A_M = \{a_1, a_2, \ldots, a_m\}. \tag{3.27}$$

Alternatives in A_M are *evaluated* with model M with function $evaluation_M$, which maps from the $inputSpace_M$ to the space of all possible evaluations $outputSpace_M$:

$$evaluation_M : inputSpace_M \mapsto outputSpace_M. \tag{3.28}$$

The evaluation of alternative a_i is done in a *bottom-up* way, using $evaluation_M(a_i)$. The input attribute values are acquired directly from a_i. The aggregation is carried out from model inputs toward its outputs according to the hierarchical structure of the model. Each aggregate value is computed using the attribute's respective aggregation function.

Algorithmically, all the aggregated attributes in model M are first *topologically sorted* with respect to S. The sorting is necessary, so that the evaluation procedure has all input values, for some aggregated attribute, readily available. The topological sorting procedure is presented in Algorithm 6.1.

The sorting determines the order of atomic aggregation function evaluations and ensures that all inputs to the current aggregation function are readily available. Then, the values of aggregated attributes are computed using the corresponding functions F_j in the given order. This produces the output evaluations corresponding to all aggregated attributes in $modelOutputs_M$. The described algorithm for evaluation is presented in Algorithm 3.1.

If all aggregation functions defined in the model are monotone increasing, then as per Section 3.2.2, completeness and consistency of evaluations can be confirmed.

Each alternative a_i is viewed as a tuple of values from $inputSpace_M$. If all functions are monotone increasing, the aggregation procedure along with statements made in Section 3.2.2 confirm the following:

$$\overrightarrow{a_i} \preceq \overrightarrow{a_j} \implies evaluation_M(\overrightarrow{a_i}) \preceq evaluation_M(\overrightarrow{a_j}). \tag{3.29}$$

Algorithm 3.1: Algorithm *evaluate* for evaluation of alternative a_i on model M.

Input: Alternative a_i, Model M
Output: $(v_j, v_k, \ldots, v_l) \in outputSpace_M$

```
1  sort = topologicalSort(M);
2  S = M.S;
3  for j = 0; j < size(M.X); j++ do
4  |  x = sort[j];
5  |  if size(S(x))>0 then
6  |  |  F_j = x.F;
7  |  |  values = [size(S(x))];
8  |  |  l = 0;
9  |  |  for Attribute x_k: S(x) do
10 |  |  |  values[l] = a_i[x_k];
11 |  |  |  l++;
12 |  |  end
13 |  |  a_i[x] = F_j(values);
14 end
15 output = [size(outputSpace_M)];
16 j = 0;
17 for Attribute x_l: outputSpace_M do
18 |  output[j] = a_i[x_l];
19 |  j++;
20 end
21 return output;
```

Additionally, if there exists an alternative $a_k \in A_M$, for which it holds:

$$\forall a_i \in A_M : evaluation_M(\overrightarrow{a_i}) \preceq evaluation_M(\overrightarrow{a_k}), \tag{3.30}$$

then a_k is the most desirable and rational alternative, and hence it should be chosen and implemented.

However in practice, there are usually multiple alternatives that in the final evaluation receive the same value. Suppose the set of alternatives $B = \{a_{k,1}, a_{k,2}, \ldots, a_{k,b}\}$ represents the alternatives that receive the same best value after the final evaluation. Then the following holds:

$$\begin{gathered} \forall b_i, b_j \in B : evaluation_M(\overrightarrow{b_i}) \approx evaluation_M(\overrightarrow{b_j}), \\ \forall a_i \in A : \forall b_j \in B : evaluation_M(\overrightarrow{a_i}) \preceq evaluation_M(\overrightarrow{b_j}). \end{gathered} \tag{3.31}$$

Identifying such sets, we can create a *sorting* of alternatives in terms of MCDA. The alternatives in DEX are sorted in at most $|D_{\text{root}}|$ classes, where *root* is one root of the considered model M. The sort is a partitioning of alternatives A_M in $|D_{\text{root}}|$ classes. Generally, the sort creates a *partial order* of alternatives — if there is a class to which more than one alternative is assigned. On the other hand, the sort may create a *total order* of alternatives if each class contains at most one alternative.

In cases when partial order is created, the DM must use approaches to find the best alternative with some other MCDA method or with numeric approaches, as described in (Mileva-Boshkoska & Bohanec, 2012).

3.3.1 Illustrative Example

Now that the whole model has been defined, alternatives can also be defined and evaluated. Let us consider three different apartments called *Big*, *Equipped* and *Nice*. Each apartment is described by a 5-tuple of values corresponding to the input attributes in the model (*buying price*, *utilities*, *location*, *interior* and *exterior*). The three apartments are shown in Table 3.5. Notice that all input values are formulated in terms of qualitative values. In reality, we determine these values from real values of alternatives. For example, we may assess the buying price of 50000 euros as medium. Since this is only an illustrative example, we assume that this process has already been carried out by the DM. The table gives an input value for the given alternative and respective input attribute. For example, apartment *Big* has *location* attribute set to *bad*.

Table 3.5: The three considered alternatives in the apartment evaluation. For each of the model input attributes, the respective alternative has a value assigned.

Name	*buying price*	*utilities*	*location*	*interior*	*exterior*
Big	medium	medium	bad	bad	ok
Equipped	medium	low	ok	ok	good
Nice	low	medium	good	bad	bad

The considered alternatives are part of the alternative set $A_{\text{apartment}}$:

$$A_{\text{apartment}} = \{\text{Big}, \text{Equipped}, \text{Nice}\}. \tag{3.32}$$

In this particular example, we see that according to Equation (3.29), a partial order of alternatives can be induced. It can be stated that input values of apartment *Equipped* dominate input values for apartment *Big*:

$$\overrightarrow{\text{Big}} \preceq \overrightarrow{\text{Equipped}}. \tag{3.33}$$

From this observation we can conclude that apartment *Equipped* will receive at least as good evaluation as apartment *Big*. Here we are also taking into account the consistency of the aggregation functions. Apartments *Equipped* and *Nice*, and *Big* and *Nice* are incomparable, since apartment *Nice* is better in some aspects and worse in other aspects than apartments *Big* and *Equipped*.

With given input values, we can evaluate the three considered alternatives. To illustrate the evaluation procedure, let us consider the alternative *Big*. A possible topological sorting of the aggregated attributes in apartment model is:

$$(\text{price}, \text{layout}, \text{apartment}). \tag{3.34}$$

Consequently, the attributes should be evaluated according to this order — from *price* to *apartment*. The value of *price* is computed by the aggregation function f_{price}, given the values for child attributes buying *price* and *utilities* (looked up directly from the alternative). The functional value for $f_{\text{price}}(\text{medium}, \text{medium})$ is *medium*, which is looked up from Table 3.3 in the fifth row. Similarly, the value for *layout* is computed by querying function f_{layout} with values (bad, bad). The produced value is *bad*. The value for the remaining aggregated attribute *apartment* is produced by looking up combination (medium, bad, bad). Here notice that the previous evaluations of computed values for *price* and *layout* were used. The apartment *Big* is assessed as *very bad*. The other two apartments, *Equipped* and

Nice, are assessed as *bad* and *good*, respectively, in the same way as *Big*. All the evaluations are shown in Table 3.6.

Table 3.6: The three considered alternatives in the apartment evaluation and the evaluations of the respective aggregated attributes.

Name	*price*	*layout*	*apartment*
Big	medium	bad	very bad
Equipped	medium	ok	bad
Nice	low	bad	good

Looking at the table of evaluations (Table 3.6), we see that apartments are sorted, where none of the apartments are evaluated with the same evaluation. Since the scale of attribute *apartment* is preferentially ordered, we can deduce that apartment *Nice* is better than *Equipped* and apartment *Equipped* is better than apartment *Big*. Here we must point out that from the fact stated in Equation (3.33), the apartment *Equipped* is indeed evaluated with a better value than apartment *Big*. Furthermore, we note that apartment *Nice* is the best considered alternative, given the model for *apartment* evaluation, according to Equation (3.30).

3.4 Software for DEX

There are currently three computer implementations of DEX method. The first is DEXi, which is the most used implementation for modeling with DEX and also the most developed. The second implementation is proDEX, which is a stand-alone implementation of DEX, which is extended with probabilities during evaluation. The third implementation is the DEXx implementation, developed in this thesis, and presented in Chapter 6.

3.4.1 DEXi

Currently, DEX method is implemented in the software called DEXi (Bohanec & Trdin, 2014; Bohanec, 2015a). DEXi supports an interactive construction of the decision model and alternatives. The software aids in defining decision rules, checking their completeness and consistency, and provides a number of analytical decision tools. The user can make an in-depth analysis of evaluations and decision alternatives. For example, for each value computed during the evaluation, there is a decision rule which was used to obtain that value — the rule explains how the evaluation was obtained. Moreover, the decision rule was applied on the basis of values computed by previous evaluations, which can also be drilled down even further, explaining the sources for such evaluation and thus providing evidence for answering the question of why the evaluation was such. The process stops when the drilling procedure reaches the input attributes and basic values of the alternative. If the DM is not satisfied with the evaluation, he/she can revisit the model structure, aggregation functions and/or the alternative's basic values.

The DEXi software includes four different analysis procedures for the evaluated alternatives (Bohanec & Trdin, 2014):

Plus-minus-1 analysis checks to which extents alternative evaluations are affected by small changes to the input attribute values,

Selective explanation informs the user about the strong and weak components of each alternative — the sub-trees where all attributes have the best (or the worst) possible values,

Compare compares the pre-selected alternatives attribute-wise, according to their values, and

Charts are able to plot k sided utility diagrams based on the selected alternatives and k selected attributes that form the diagrams.

In Figure 3.1 we present DEXi software which has the model from the illustrative example loaded. On the left side, the figure shows the structure of the apartment model. The right part shows the properties of the *apartment* attribute, such as name, description, scale and the aggregation function.

Figure 3.2 shows the function definition of the *apartment* attribute. The three leftmost columns correspond to the three input attributes: *price*, *location* and *layout*. In the three columns all possible combinations of input values are generated. The fourth column gives the output value of the aggregation function, for the respective row. The fourth column is filled-in by the DM. The upper part of the figure gives access to additional modeling options, such as automatic generation of output values with predefined weights. Furthermore, the lower part of the image presents various statistics regarding the output values.

Figure 3.3 shows the three selected alternatives from the illustrative example in DEXi. The alternatives are shown with their name in a tabular form, where the left-most column presents the structuring of the attributes. The three right columns give the computed aggregated values (together with the final evaluation) and their respective input values.

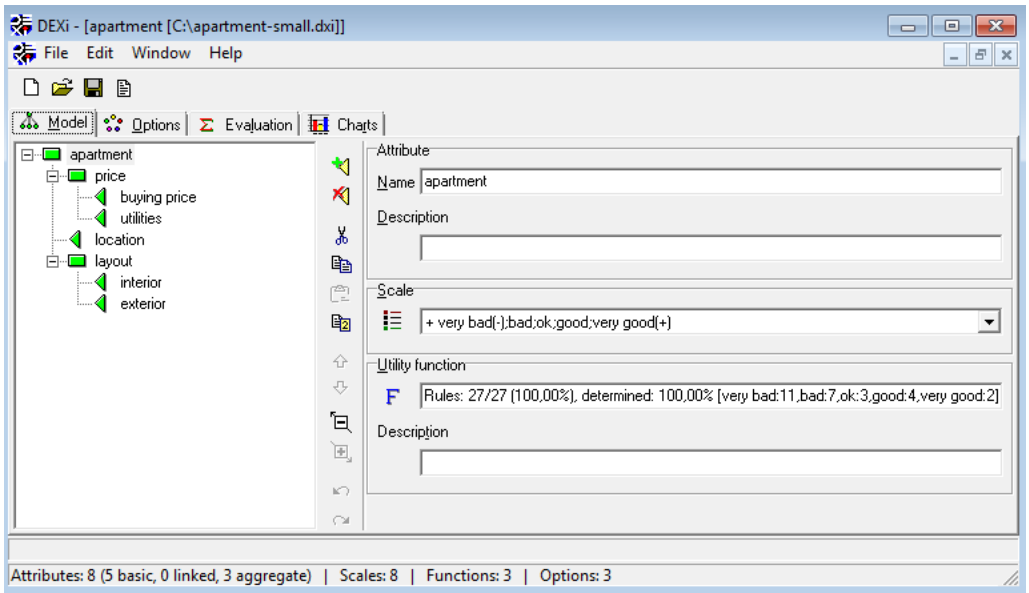

Figure 3.1: Screen capture from DEXi software. The left part shows the tree structure for the model used in the illustrative example. The right part shows the properties of the *apartment* attribute such as its name, description, scale and aggregation function.

3.4.2 proDEX

proDEX (Žnidaršič et al., 2006a, 2006b; Žnidaršič, 2015) is a stand-alone Python implementation of DEX, which is extended with probabilities in alternative values and probabil-

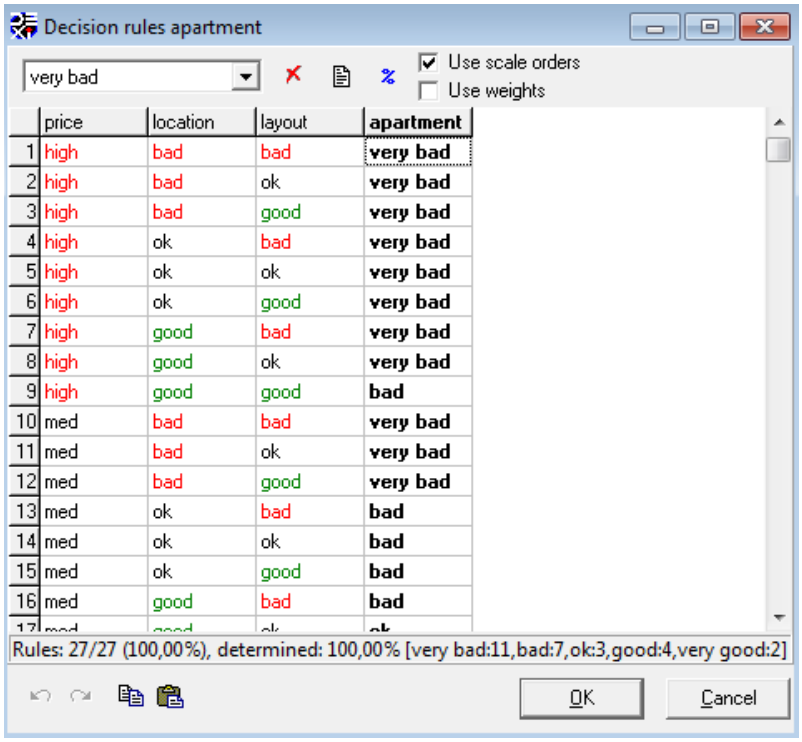

Figure 3.2: Screen capture from DEXi software, while defining an aggregation function for *apartment* attribute. The three leftmost columns give all possible value combinations for the input attributes *price*, *location* and *layout*. The fourth column gives the output value for the respective row (input values).

Option	Big	Equip	Nice
. apartment	very bad	bad	good
. . price	med	med	low
. . . buying price	med	med	low
. . . utilities	med	low	med
. . location	bad	ok	good
. . layout	bad	ok	bad
. . . interior	bad	ok	bad
. . . exterior	ok	good	bad

Figure 3.3: The figure shows the three considered alternatives in the illustrative example, with their respective input values and their evaluations of aggregated attributes.

ities in function outputs. The implementation additionally supports model revision, which enables the online model updates, given already evaluated alternatives. The model is updated by manipulating the *confidence* of function output values and their corresponding probabilities.

proDEX is mainly run from the terminal. However, an additional graphical user interface (GUI) was also developed (Žnidaršič, 2015) for usage of proDEX in Orange suite (Demšar et al., 2013) for experimental machine learning and data mining.

Chapter 4

Extending DEX

The many uses of DEX have indicated a great practical value of the method but have also revealed the need to extend it in several directions. Five substantial extensions (Trdin & Bohanec, 2012, 2014a) were identified by practical needs, especially with the aim to provide additional features for DMs.

In this chapter we extend the DEX methodology with features that are currently missing and are needed, so that the methodology will be able to cope with increasing demands of decision makers and their decisions. The DEX method is formally extended into five directions, where two of the advances are unified as one — for the methodology to work, both advances need to be included.

- Native support for *full hierarchies* of attributes, since currently the hierarchies are rigidly handled by the so-called links.
- *Numerical attributes* will be incorporated into the methodology in order to support decision problems that are composed of qualitative and numeric data. With that, data preprocessing and transformation will be eliminated. The extension of numerical attributes also requires to aggregate different value types. For that purpose we extend DEX into direction of *general aggregation functions*, where the DM is able to define a wide variety of functions based on numeric and qualitative inputs.
- Simple qualitative and quantitative values are extended to include complex data, such as fuzzy sets, distributions, sets and intervals of values. The use of *extended domains* also needs to propagate the possible probabilities of values and membership values of fuzzy sets to the higher-level attributes.
- *Relational aggregation* of alternatives, in order to support decision problems dealing with relational alternatives, that is alternatives composed of several sub-components.

The newly developed method, based on DEX, which also includes the extensions presented in this thesis, is called *DEXx*.

4.1 Full Hierarchies

Regarding the structure of the DEX model, the needed generalization is using full hierarchies or acyclic directed graphs — DAGs, instead of trees. Usage of any other more general graph structure is not feasible, since evaluation of alternatives is not possible with the introduction of cycles. A basic example of contradiction is with two attributes, which both depend on each other. Evaluation of the first needs the value of the second, but the value of the second can be computed only if it had the value of the first.

In principle, the structure of the DEX model is a *hierarchy*, i.e. *directed acyclic graph*. So far, hierarchies are only indirectly supported in DEX (Bohanec, 2015a; Bohanec & Rajkovič, 1990; Bohanec, Rajkovič, et al., 2013) and DEXi software (Bohanec, 2015b, 2014), using the concepts called "chaining" and "linking" of nodes. In DEXi, the edges are called links. Links are created automatically, if the user defines two attributes with the same name and a same scale. The functionality is great for usage with not many linked attributes, but with more complex models with lots of links, the model may become cluttered and heavy to read. Linking also affects structuring of the model — for instance (Craheix et al., 2015) suggest to separate linked attributes from the main tree.

In the extension we introduce the support for native hierarchies by representing the model using a direct graph form. Hierarchies also natively support multiple root attributes.

4.1.1 Extension Formalization

The full hierarchies extension does not need any additional formalization, to include it into DEXx. Recall, in Equation (3.4) S describes a mapping from attribute x, to a set of attribute's descendants $\{x_i, x_j, \ldots, x_l\}$. The mapping does not prohibit the usage of S, to construct the model as a hierarchy.

Additionally, we can define equations that check if the represented model is a tree or a hierarchy.

In case when a model represents a tree, there are no two attributes, which have a same child attribute. As a condition, this can be stated as:

$$\bigcup_{x,y \in X, x \neq y} S(x) \cap S(y) = \emptyset. \tag{4.1}$$

In the second case when a model represents a hierarchy of attributes, there must exist at least one pair of different attributes $x, y \in X$, which have a child attribute in common. As a condition, this can be stated as:

$$\bigcup_{x,y \in X, x \neq y} S(x) \cap S(y) = \{x_{h_1}, x_{h_2}, \ldots, x_{h_H}\} \neq \emptyset. \tag{4.2}$$

Here, $\{x_{h_1}, x_{h_2} \ldots, x_{h_H}\}$ represents the set of all attributes that have more than one parent attribute — we sometimes call them *hierarchical attributes*. If these attributes were removed from the attribute hierarchy, the model would be decomposed into several connected components, where each component would form a tree. Notice that Algorithm 3.1 already supports hierarchical models by employing topological sorting.

4.2 Numeric Attributes and General Aggregation Functions

Most MCDM methods are *quantitative* — they involve numeric attributes and real-valued utility functions. DEX is essentially a qualitative method and currently uses only qualitative attributes. This approach is not always appropriate because it requires that quantitative values are discretized before use, even for quantities that are naturally measured and expressed with numbers. Furthermore, in many decision problems there is a need to support both qualitative and quantitative attributes within the same model (Bohanec et al., 2013, 2014; B. Kontić, Bohanec, Trdin, et al., 2014; B. Kontić et al., 2016). Even though a few quantitative features have already been considered within DEX (Mileva-Boshkoska & Bohanec, 2012; Trdin & Bohanec, 2012, 2014a, 2014b; Žnidaršič, Bohanec, & Bratko, 2003), there is a strong need to systematically introduce numerical attributes into the DEX method.

Adding numerical attributes requires a number of representational and algorithmic extensions, such as adding numerical aggregation functions and handling transformations between qualitative and numeric values. Attribute value scales have to be extended to include *real* numbers, *integer* numbers, *finite intervals over* real numbers and *finite integer intervals*. Also, while being edited, models should be able to transform their internal data, possibly avoiding excessive user interaction.

Supporting numeric attributes does not affect the principle of structuring attributes in a hierarchy, and assigning aggregation functions to aggregated attributes. It however affects the definition of domains of attributes.

There is also an essential difference in the aggregation functions themselves because, in general, they have to cope with various combinations of qualitative and quantitative attributes, both at function inputs (arguments) and function outcomes. For this reason, we consider six different function types, which differ by value types of input attributes and the output attribute.

There are three combinations of basic types the function can receive: (1) all qualitative values (denoted QQ), (2) all quantitative (numeric) values (NN) or (3) a mix of quantitative and qualitative values (NQ). Regarding the output, the function can yield only one type of output — either qualitative (Q) or quantitative (N). Note that the basic DEX method covers only one of these cases: functions that map from qualitative arguments to qualitative values (denoted $QQ \mapsto Q$).

The introduction of quantitative attribute values increases the expressiveness of attributes' value scales and the aggregation of such quantitative values. The extension introduces a natural way to incorporate numeric values and operations. Inevitably, the drawback of the extension is an increased complexity due to introduction of new types of aggregation functions.

4.2.1 Extension Formalization

First, we need to extend the definition of the value scale D given in Equation (3.2) and Equation (3.3) to include numeric quantities:

$$D_i = \begin{cases} (w_{i,1}, w_{i,2}, \ldots, w_{i,m_i}) & \\ [a, b] & a, b \in \mathbb{Z} \cup \{-\infty, \infty\} \\ [c, d] & c, d \in \mathbb{R} \cup \{-\infty, \infty\} \end{cases} . \tag{4.3}$$

This equation represents three possibilities for defining the value scale D_i:

1. The first option remains the same as before: the DM may choose a finite list of *words*. The list may be preferentially ordered.

2. The second option defines an *integer interval* by specifying two integer values $a, b, a \leq b$. The scale of the particular attribute is then composed of all integers between a and b, inclusively. Note that we can also specify an unbounded interval using negative or positive infinities.

3. The third option defines a *real interval* by choosing two real values $c, d, c \leq d$. The possible values of such attribute are all real values between c and d, inclusively. Both infinities are also allowed.

To model all these situations, a general framework is needed, which would be able to produce either numeric or qualitative values, given any combination of numeric and/or

qualitative attributes, and take into account the DMs preferential knowledge about aggregation functions and model structure. We are not aware of any such framework, therefore we propose the following one.

The three cases of possible value scale types cover six types of *aggregation functions*, however in the extension only five *primitive aggregation functions* are considered (Figure 4.1, 4.2, 4.3, 4.4 and 4.5). Here, "primitive" means that the function can be defined in "one step", using either a single formula or a single decision table. In contrast, a "non-primitive" or "compound" aggregation is defined by a hierarchical composition of primitive functions. It is very important to formalize all possible primitive aggregation functions, for three reasons. The first and foremost, to avoid any confusion in using functions. The second reason for formalization is to provide the users of functions with same definitions — which guarantees that functions will always produce the same results. The third reason is that formalized functions provide guarantees for their input and output types, which incidentally means that the output of the function (and the whole model) is well defined.

Only five primitive aggregation functions are considered, because they are sufficient for representation of other compound functions.

Function type 1 ($QQ \mapsto Q$): These functions have only qualitative inputs and a qualitative output. They are already covered by the current DEX method and modeled by decision rules according to Equation (3.17).

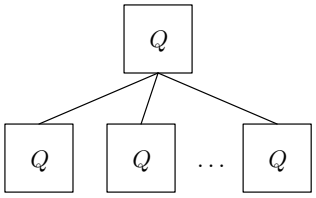

Figure 4.1: Graphical representation of *Function type 1*.

Function type 2 ($QQ \mapsto N$): These functions have only qualitative inputs and a numeric output. They are modeled similarly to Function type 1 by decision rules — the only difference is that they output a single number or numeric interval instead of a single word or interval of words. The output number is restricted to the scale of the aggregated attribute. Function type 2 is already formally covered by Equation (3.17), taking the extended definition of D_i in Equation (4.3). Note that this function type includes the special case $Q \mapsto N$, where only one qualitative attribute is converted to a numerical attribute.

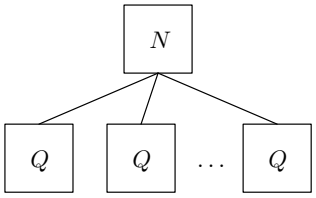

Figure 4.2: Graphical representation of *Function type 2*.

Function type 3 ($NN \mapsto N$): These functions have only numeric inputs and a numeric output. They are modeled in a purely mathematical way using mathematical operators on numeric functions' arguments, such as $+$, $-$, $\times$, $/$, power, square, root, etc., and numeric constants. A common representative of this function type is the weighted average — for

example, $\sum_{i=1}^{n} w_i v_i$ — where w_i is the i-th attribute's weight and v_i is its value. Similar to Function type 2, this function type is already formally covered by Equation (3.17). The difference is that all $D_j, D_k, \ldots, D_l, D_i$ are numeric scales, and function f_i is essentially a mathematical expression. Functions of this type are commonly used in quantitative MCDA methods and can be thus easily adopted in DEX at this point, together with already developed support for their acquisition and use in practice. For instance, for Function type $NN \mapsto N$, the DM can use the weighted sum and employ the pairwise comparison approach of AHP (Saaty, 2008) to elicit attribute weights.

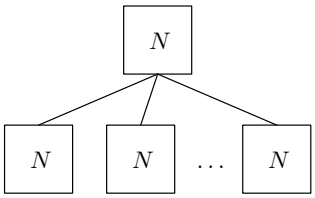

Figure 4.3: Graphical representation of *Function type 3*.

Function type 4 ($N \mapsto Q$): These functions discretize a single numeric input attribute to an output qualitative attribute. Generally, a discretization function of the i-th attribute (NQ_i) maps real numbers to qualitative value scales:

$$NQ_i : \mathbb{R} \mapsto D_i. \tag{4.4}$$

$$NQ_i(y) = \begin{cases} w_p & \text{if } y \in [a, b] \\ \ldots & \\ w_r & \text{if } y \in [c, d] \end{cases} \tag{4.5}$$

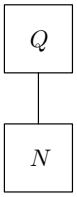

Figure 4.4: Graphical representation of *Function type 4*.

Function type 5 ($NQ \mapsto N$): These functions have qualitative and numeric inputs and a numeric output. The primitive specification of such functions is based on decision tables, similar to function types 1 and 2. First, we define a decision table using all the qualitative function arguments. Second, in each row of the table we define the numerical function of the remaining numeric attributes. Formally, suppose that $q_1, q_2, \ldots, q_k$ are all qualitative inputs of x_i and $n_1, n_2, \ldots, n_m$ are all numeric inputs of x_i. Let $\Phi(x_i)$ denote the space of functions which receive all x_i's numeric inputs and output a numeric value from D_i:

$$\Phi(x_i) = \{f | f : D_{n_1} \times D_{n_2} \times \cdots \times D_{n_m} \mapsto D_i\}. \tag{4.6}$$

Let h_i be a function which maps from all combinations of qualitative inputs to space $\Phi(x_i)$:

$$h_i : D_{q_1} \times D_{q_2} \times \cdots \times D_{q_k} \mapsto \Phi(x_i). \tag{4.7}$$

The overall aggregation function f_i is then evaluated in two steps: (1) the value of $h_i(q_1, q_2, \ldots, q_k)$ is used as a table look-up index to find the corresponding function $g \in \Phi(x_i)$, and (2) evaluate $g(n_1, n_2, \ldots, n_m)$. Formally:

$$f_i(q_1, q_2, \ldots, q_k, n_1, n_2, \ldots, n_m) = h_i(q_1, q_2, \ldots, q_k)(n_1, n_2, \ldots, n_m). \tag{4.8}$$

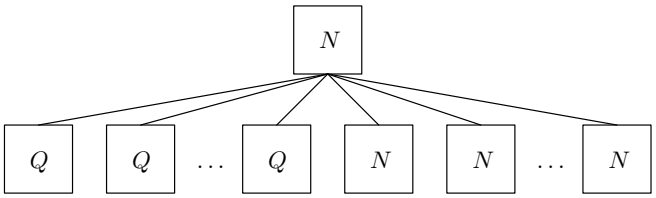

Figure 4.5: Graphical representation of *Function type 5*.

We also considered two different formalizations of *function type 5*, but both formalizations can be be expressed using other primitive function types, removing the expressive power of the presented *function type 5* — hence the function would not be primitive any more. The two other considered formalizations are:

- modeling the function by firstly discretizing numeric values to qualitative values (using *function type 4*) and finally mapping all qualitative values to a numeric output (using *function type 2*). The function expression is $(N \mapsto Q)Q \mapsto N$.
- The second option considered firstly maps qualitative values to a numeric value (using *function type 2*) and mapping all numeric values to a numeric value by a mathematical expression (using *function type 3*). Expression for such definition is $N(Q \mapsto N) \mapsto N$.

In machine learning terms, *Function type 5* resembles one *leaf* in a *model tree* (Frank, Wang, Inglis, Holmes, & Witten, 1998; Landwehr, Hall, & Eibe, 2005). Similarly to model tree leafs, *Function type 5* outputs a numeric function value, based on the split variables. $NQ \mapsto N$ function's inputs represent the corresponding conditions in the split of the model tree, and the functions in the outputs correspond to the functions in the leaves of the model tree. Note that similar deduction can be made for function $QQ \mapsto N$.

Using the five primitive function types, we can express other (compound) aggregations of numeric and qualitative values. Among these, two are of special interest: $NN \mapsto Q$ and $NQ \mapsto Q$.

Compound function type $NN \mapsto Q$: These compound functions have only numeric inputs and a qualitative output. They can be modeled in two ways:

1. Each numeric attribute can be first discretized and then aggregated by *Function type 1*: $(N \mapsto Q)(N \mapsto Q) \mapsto Q$.
2. Since the arguments are all numeric, a mathematical expression can be constructed by *Function type 3*, and then a single discretization can be applied on the output: $NN \mapsto N \mapsto Q$.

Compound function type $NQ \mapsto Q$: This function type has qualitative and numeric inputs and a numeric output. It can be modeled with primitive function types in three ways:

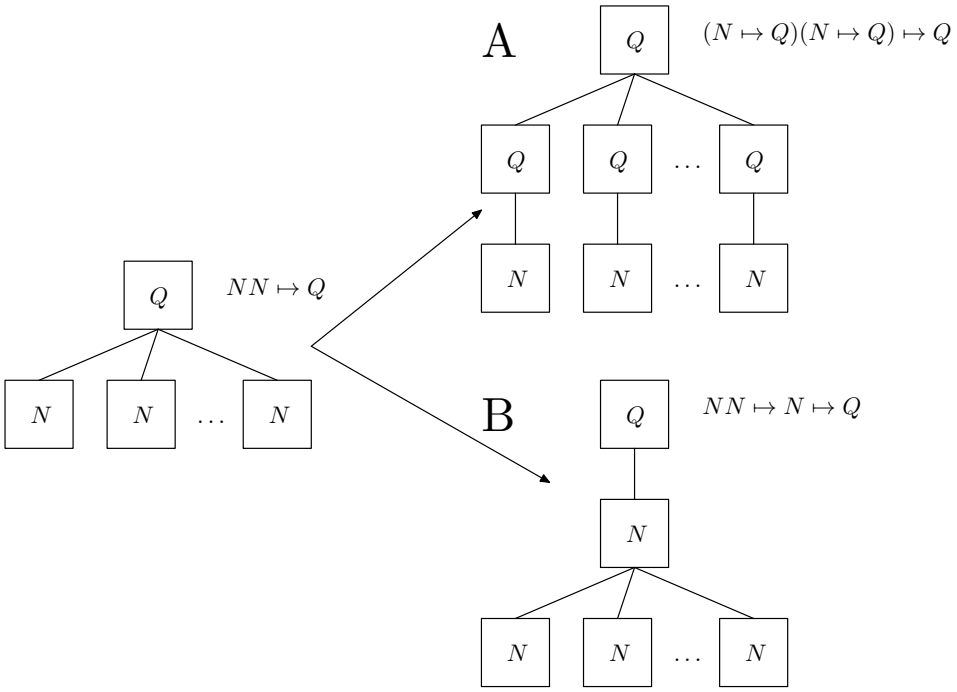

Figure 4.6: Two possibilities for modeling Compound function type $NN \mapsto Q$. Option A suggests that numeric attributes are first discretized to qualitative and then the function is modeled as a rule based qualitative function. Option B indicates that numeric attributes are aggregated using Function type 3, and then discretizing the produced result.

1. Discretizing the result retrieved from a *Function type 5*: $(NQ \mapsto N) \mapsto Q$.
2. Discretizing numeric inputs and creating a qualitative function on all qualitative attributes: $(N \mapsto Q)Q \mapsto Q$.
3. Using *Function type 3* on numeric attributes, discretizing its output and creating a qualitative function on the result and qualitative attributes: $((NN \mapsto N) \mapsto Q)Q \mapsto Q$.

The two compound function types are correspondingly presented graphically according to their descriptions in Figure 4.6 and Figure 4.7.

The compound function types $NN \mapsto Q$ and $NQ \mapsto Q$ can thus be modeled in a number of different ways. On one hand, this is convenient, as the functions can be flexibly adapted to the characteristics of the problem. On the other hand, this puts additional burden on DM, as he/she has to decide which option to use in a given situation. This decision is important, as different compound functions generally lead to different results. The proposed approach fulfills the requirement for flexibility, but does not answer the question on how to decide which option should be used. This is left for further research and practice. At the time being, we assume that the choice of compound functions is entirely at the DM's responsibility.

4.3 Probabilistic and Fuzzy Distributions

In its current form, DEX evaluates alternatives with *crisp* values. The only exception is in cases when a decision table yields an interval of values — in this case, the resulting

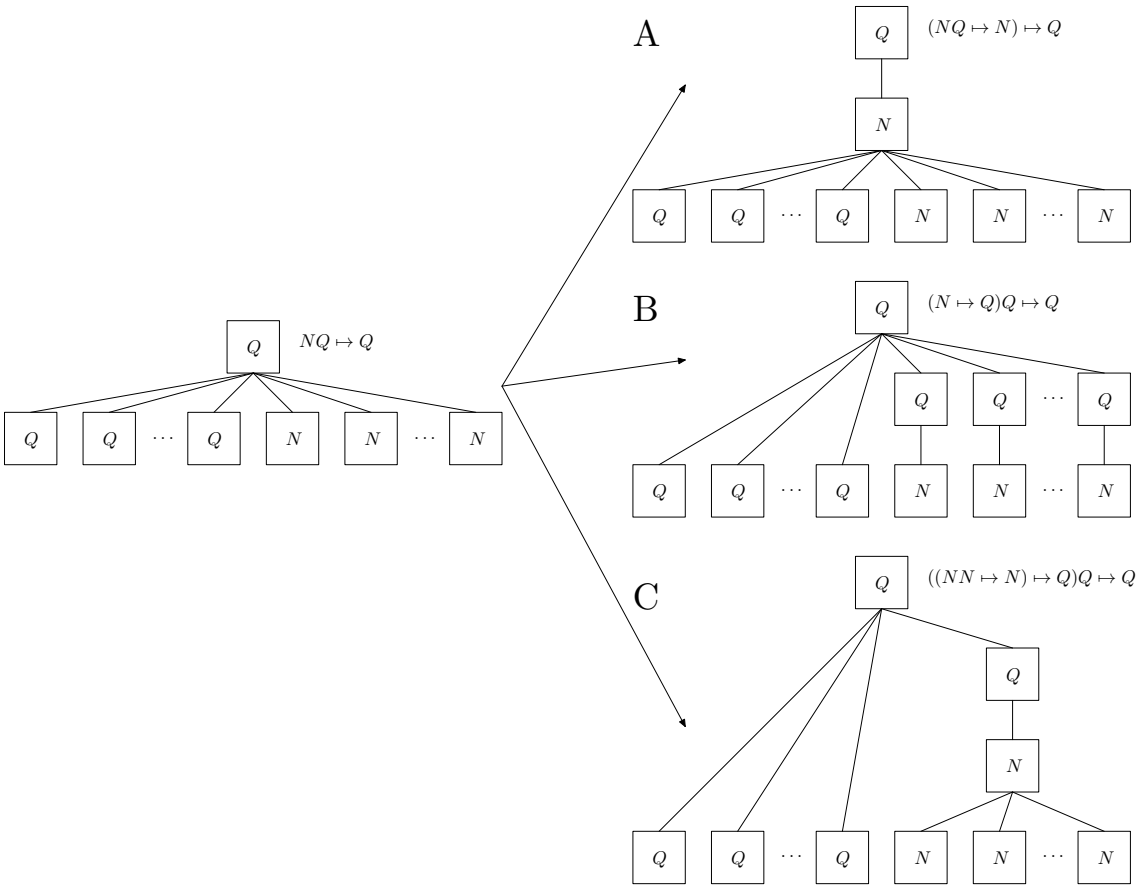

Figure 4.7: Three possibilities for modeling Compound function type $NQ \mapsto Q$. Option A gives the configuration to use Function type 5 on the inputs and then discretize the computed output. Option B gives the possibility to discretize the numeric inputs and creates a qualitative function on outputs and remaining qualitative inputs. Option C presents the possibility to aggregate numeric attributes to single numeric value, discretize the value and then construct a qualitative function based on this and on qualitative inputs.

evaluation is generally a set of values. However, alternatives and the DM's preferences are often *imprecise* or *uncertain*. For instance, alternatives' values may be difficult to measure or assess precisely because they may depend on factors that cannot be controlled by the DM. Similarly, sometimes it is difficult for the DM to define a decision rule that would produce a single crisp value because the outcome is distributed between several values. Therefore, we propose an extension to DEX that explicitly models these types of uncertainty by introducing probabilistic or fuzzy distributions of values.

Probabilistic distributions were introduced to aggregation functions in DEX and implemented in the system called proDEX (Žnidaršič et al., 2006b), in order to facilitate the revision of already developed DEX models (Žnidaršič et al., 2006a). Probabilistic distributions were also considered in HINT (Hierarchy INduction Tool) (Zupan et al., 1999) for usage in function decomposition in machine learning (Zupan et al., 2001).

In this section we propose a systematic extension to DEX which introduces probabilistic and fuzzy value distributions to both alternative values and aggregation functions, taking into account the formal model described in Chapter 3 and the other extensions proposed in this thesis. The distributions are introduced at two points:

1. Values describing decision alternatives are extended to probabilistic or fuzzy distributions of values over corresponding attribute scales, and

2. outputs of aggregation functions are extended from crisp values or intervals to distributions of values.

That is, wherever we could have previously used a crisp value or an interval of values in a model, we may now use a distribution of values. This requires three changes to the formal model: (1) attribute scales have to be extended to cope with value distributions, (2) aggregation functions should in general map to distributions rather than intervals and (3) the evaluation procedure should propagate distributed values.

We also considered adding higher-order uncertainty to the value distributions, such as *confidence* into the computations. With such a value added to each value distribution, we could acquire DM's confidence in the value assigned to an alternative or confidence into correctness or appropriateness of evaluation. The notion of higher-order uncertainty was included in proDEX (Žnidaršič et al., 2006b), where it was extensively used for model revision. For simplicity, higher-order uncertainties were not considered in the extended DEX methodology so far.

Value distributions are general in the sense that they are capable of representing all previous value types: single crisp values, value intervals and value sets alongside the newly required probabilistic and fuzzy distributions of values. Previously defined aggregation functions, which did not include distributions, can thus be used without change in the new setting.

4.3.1 Prerequisites for Formalization

In order to formally introduce the value distributions into the methodology, we need to give some mathematical prerequisites.

Given an attribute x and its scale D_x, we define a value distribution V as a tuple (S, r), where $S \subseteq D_x$ and $r : S \mapsto [0, 1]$. For finite S, we use the notation $V = \{v_1/p_1, v_2/p_2, \ldots, v_k/p_k\}$, where $p_i = r(v_i)$ for each i. Generally, we do not impose any specific interpretation on p_i — however, we are interested in two specific cases:

1. V is a *discrete probability distribution*: here, p_is represent probabilities of corresponding elements v_i, and are normalized so that $\sum_{i=1}^{k} p_i = 1$.

2. V is a *fuzzy set* (Zadeh, 1965): p_is represent fuzzy grades of membership of corresponding elements v_i. Moreover, a fuzzy set is said to be normalized when $\max_{i=1}^{k} p_i = 1$.

In this way, we can represent all needed value types using both representations:

- A crisp value v can be represented as a value distribution $\{v/1\}$.
- A qualitative interval of values $[v_1, v_k]$, which includes all values $v_1, v_2, \ldots, v_k$, is represented as a discrete probability distribution $\{v_1/\frac{1}{k}, v_2/\frac{1}{k}, \ldots, v_k/\frac{1}{k}\}$ or, alternatively, as a fuzzy set $\{v_1/1, v_2/1, \ldots, v_k/1\}$.
- The set of values $v_1, v_2, \ldots, v_k$ is represented as a discrete probability distribution $\{v_1/\frac{1}{k}, v_2/\frac{1}{k}, \ldots, v_k/\frac{1}{k}\}$ or as a fuzzy set $\{v_1/1, v_2/1, \ldots, v_k/1\}$.

For infinite S, we treat an attribute x as a random variable X and consider the probability distributions of its values. When X is a continuous random variable, we assume the existence of probability density function (Feller, 1968) $f_X : D_x \mapsto [0, 1]$, so that

$$P[a \leq X \leq b] = \int_a^b f_X(t)dt, \tag{4.9}$$

where $a, b \in D_x$.

When X is a discrete random variable, we assume the existence of probability mass function (Feller, 1968) $f_X : D_x \mapsto [0, 1]$ so that

$$f_X(t) = P(X = t) \tag{4.10}$$

where $t \in D_x$.

To formally describe the evaluation procedure using value distributions, we need to transform all value distributions to a common representation. When interpreted as probability distributions, discrete and continuous variables can be treated in a unified way. In order to represent a discrete random variable with a probability density function, we employ the *Dirac delta function* (Hewitt & Stromberg, 1965):

$$\delta(x) = \begin{cases} \infty & \text{if } x = 0 \\ 0 & \text{if } x \neq 0 \end{cases}. \tag{4.11}$$

If a discrete random variable takes values $v_1, v_2, \ldots, v_k$ with probabilities $p_1, p_2, \ldots, p_k$, then its associated probability density function is:

$$f(t) = \sum_{i=1}^{k} \delta(t - v_i). \tag{4.12}$$

Suppose there are n independent random variables X_i, $i = 1, \ldots, n$ which are associated with probability density functions $f_{X_i}(x_i)$. Let $Y = G(X_1, X_2, \ldots, X_n)$ be a random variable which is a function of X_is. Then the density function $f_Y(y)$ is (Feller, 1968; Hewitt & Stromberg, 1965):

$$f_Y(y) = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} f_{X_1}(x_1) f_{X_2}(x_2) \ldots f_{X_n}(x_n) \delta(y - G(x_1, x_2, \ldots, x_n)) \, \mathrm{d}x_1 \mathrm{d}x_2 \ldots, \mathrm{d}x_n. \tag{4.13}$$

In general, solving this multiple integral is a difficult problem to tackle analytically, thus we suggest employing statistical sampling with the *Monte Carlo* method (Caflisch, 1998) for all random variables defined with a probability density function: each random variable X_i is sampled m times, giving a finite discrete probability distribution $V_i = \{v_j/\frac{1}{m} | v_j \in X_i, j = 1, 2, \ldots, m\}$.

Now, we can consider only value distributions with a finite number of elements for probabilistic aggregation. Given some aggregation function g, we need to compute the value distribution $W = g(V_1, V_2, \ldots, V_n)$, where each distribution V_i is obtained either by sampling or as a result of computation on lower levels of the attribute hierarchy. Also note that function g is defined only on atomic crisp values, see Equation (3.17). In order to obtain $W = \{w_1/p_1, w_2/p_2, \ldots, w_k/p_k\}$, where $w_i \in D_Y$, we need to compute all $p_i, i = 1, 2, \ldots, k$:

$$p_i = \sum_{(v_1/q_1, v_2/q_2, \ldots, v_n/q_n) \in V_1 \times V_2 \times \cdots \times V_n} \Big(\prod_{w_i/r_i \in g(v_1, v_2, \ldots, v_n)} r_i \quad (\prod_{j=1}^{n} q_j) \Big). \tag{4.14}$$

This equation produces a probability for value w_i in the final value distribution W. The outer sum runs over the Cartesian product of all basic values in input value distributions. Given some value combination $v_1, v_2, \ldots, v_n$, the basic function $g(v_1, v_2, \ldots, v_n)$ is

computed. The result is generally a value distribution $\{w_1/r_1, w_2/r_2, \ldots, w_k/r_k\}$. From this distribution, only the probability r_i of w_i is considered. This particular combination adds to the outer sum the probability of the output r_i multiplied by the probability of this value combination ($\prod_{j=1}^{n} q_j$). This procedure follows the intuitive explanation of handling probabilities. Furthermore, Equation (4.14) is in line with the constraint imposed in Equation (3.21), specifying that the function F_i should for singleton input value distributions produce the same value as the function f_i.

Fuzzy sets are handled in essentially the same way, except that product and sum operators are replaced by the standard fuzzy set operators: *t-norms* ($\top$) as logical conjunction and the corresponding *t-conorms* ($\perp$) as logical disjunction (Bede, 2012):

$$p_i = \perp_{(v_1/q_1, v_2/q_2, \ldots, v_n/q_n) \in V_1 \times V_2 \times \cdots \times V_n} (0, \top_{w_i/r_i \in g(v_1, v_2, \ldots, v_n)} (r_i, \top(q_1, q_2, \ldots, q_n))). \quad (4.15)$$

The most commonly used t-norms and t-conorms in fuzzy sets are minimum and maximum, respectively:

$$\top(a, b) = \min(a, b), \quad (4.16)$$

$$\perp(a, b) = \max(a, b). \quad (4.17)$$

By substituting the t-norm and t-conorm, we get an equation for computing p_is for fuzzy sets:

$$p_i = \max_{(v_1/q_1, v_2/q_2, \ldots, v_n/q_n) \in V_1 \times V_2 \times \cdots \times V_n} (0, \min_{w_i/r_i \in g(v_1, v_2, \ldots, v_n)} (r_i, \min(q_1, q_2, \ldots, q_n))). \quad (4.18)$$

Even though Equation (4.15) is general and allows the use of any t-norm or t-conorm, the current implementation supports only probabilistic and standard fuzzy set operators.

4.3.2 Extension Formalization

In order to formally extend DEX to handle value distributions, we first need to extend the scales of attributes. For practical reasons, we wish to keep the definition of value scale D_i of attribute $x_i \in X$ the same as before. However, we extend the range of aggregation functions to include value distributions over D_i and accommodate the propagation of value distributions during evaluation. Thus, we define ED_i as the space of all value distributions over D_i. When D_i is a finite discrete scale $D_i = \{v_1, \ldots, v_k\}$, then

$$ED_i = \{\{v_1/p_1, v_2/p_2, \ldots, v_k/p_k\} | p_j \in [0, 1], v_j \in D_i, j = 1, 2, \ldots, k\}. \quad (4.19)$$

When D_i is infinite, then $ED_i = X_i$ is a random variable with an associated probability density or probability mass function $f(x_i)$ as defined in Equations (4.9) and (4.10).

Then, we need to extend Equation (3.17) so that each aggregation function f_i can, in general, map to extended domain ED_i instead of the space of intervals $I(D_i)$:

$$f_i : D_j \times D_k \times \cdots \times D_l \mapsto ED_i, \{x_j, x_k, \ldots, x_l\} = S(x_i). \quad (4.20)$$

This substitution affects the aggregation procedure carried out at each aggregated attribute x_i. When evaluating the associated aggregation function f_i, we should expect value distributions at inputs. Also, the output result will, in general, be a value distribution, too. Therefore, we have to redefine the function F_i, defined in Equation (3.20) to

$$F_i : ED_j \times ED_k \times \cdots \times ED_l \mapsto ED_i, \tag{4.21}$$

where $\{x_j, x_k, \ldots, x_l\} = S(x_i)$.

When encountering a finite value distribution from some scale, the evaluation must propagate the corresponding probabilities or membership values of particular values to construct the final evaluation. Even though the scales of attributes have been extended, the aggregation functions f_i stay the same — only the evaluation procedure needs to be adapted to handle the new extended domains. The adaptation defines how the aggregation is performed for function F_i, using function f_i. In Section 4.3.1, we suggested handling only finite value distributions since infinite value distributions are sampled.

We suggest using two types of aggregation: probabilistic aggregation or fuzzy aggregation. We leave the choice to the user of the model and assume that the type of aggregation has been defined in advance. We also assume that all value distributions involved in the aggregation are normalized accordingly — that is, $\sum_{i=1}^{k} p_i = 1$ for probabilistic aggregation and $\max_{i=1}^{k} p_i = 1$ for fuzzy aggregation with normalization. The computation of F_i is performed according to Equation (4.14) for probabilistic aggregation and according to Equation (4.18) for fuzzy aggregation. In both cases, the obtained result is a value distribution which can be interpreted as a probability or fuzzy distribution, respectively.

4.3.3 Simplification of Extended Domain Values

For practical reasons, we introduce the final step that simplifies the obtained value distributions so that they are more readable for the user. The procedure is called simplify: at input, it takes the full representation of a value distribution $V = \{v_1/p_1, v_2/p_2, \ldots, v_n/p_n\}$ and produces an equivalent simplified representation. It detects whether V represents some special distribution, such as a single crisp value, an interval or a set. Formally, $\text{simplify}_i(V)$, applied on the value distribution V assigned to attribute x_i, is a recursive procedure presented in Algorithm 4.1.

Algorithm 4.1: Algorithm for simplification of extended domain values.

Input: Value $V \in ED_{x_i}, V = \{v_1/p_1, v_2/p_2, \ldots, v_n/p_n\}$
Output: Simplified value $\in ED_{x_i}$

```
1 if p_1 = p_2 = ··· = p_n then
2 |   return simplify({v_1, v_2, ..., v_n});
3 else if V = {v_1, v_2, ..., v_n} ∧ D_i ∩ V = [v_l, v_h] then
4 |   return simplify([v_l, v_h]);
5 else if V = [v_l, v_h] ∧ v_l = v_h then
6 |   return v_l;
7 else
8 |   return V;
```

Here, the first case detects that all p_is are equal, therefore V represents at least a set. The second case determines whether or not V is an interval, and the third case checks whether an interval has the same bounds and hence represents a single value.

4.3.4 Stochastic Dominance on Extended Domain

When using value distributions, final evaluations cannot always be compared directly. One of the approaches is to employ stochastic dominance (Hadar & Rusell, 1969) on probabilistic value distributions in order to determine which value distribution is preferentially better.

Suppose we are given two value distributions $V_1, V_2 \in ED_x$, where D_x is a preferentially ordered scale. We say that V_1 (stochastically) dominates V_2 if and only if:

$$\begin{aligned} &\forall w \in D_x : P[V_1 \geq w] \geq P[V_2 \geq w], \\ &\exists w \in D_x : P[V_1 \geq w] > P[V_2 \geq w]. \end{aligned} \tag{4.22}$$

Stochastic dominance can bring some insight into the final evaluations — however, it cannot always generate a total order of alternatives. This means that the best alternative may not exist for the current model and input values of the evaluated alternatives. In such cases, the DM must interpret the results, and chose the best alternative among those that are not dominated.

4.4 Relational Models

The data encountered in everyday life are frequently of a relational nature in the sense that one entity is composed of several similar *sub-entities* — similar to the extent that they can be assessed or evaluated by the same criteria. For example, when evaluating a company, a DM may want to evaluate all departments of that particular company. Here the company's departments are the similar sub-entities, and there is a "one-to-many" relationship between the company and its departments. All departments can be evaluated in a similar way — by the same model. The problem, however, is that an *arbitrary* number of *sub-evaluations* are acquired in this way, which need to be *combined* in the evaluation of the main alternative (company).

Currently, relational models are not supported in the DEX method. Adding them would be a substantial improvement, which would facilitate addressing a much larger group of decision problems. Introducing relational models, however, requires extensions that affect the representation of decision alternatives and introduces new components, such as relational attributes and relational aggregation functions. The main purpose of this section is to extend the existing DEX method for the support of *relational models* and the *relational aggregation* of alternatives.

We propose to extend DEX to handle situations where one alternative is composed of several sub-alternatives (see examples in (Bohanec et al., 2013, 2014; Trdin & Bohanec, 2012, 2013, 2014a, 2014b)). For this purpose, we propose to modify the modeling process by developing two models: one (M) for the evaluation of the main alternative and another (RM) for the evaluation of relational sub-alternatives. To evaluate a single main alternative, each sub-alternative is first evaluated by RM. Then, all evaluations are aggregated, providing an input value to M. The evaluations in both M and RM are carried out in the same way as described in the previous sections — the only differences occur at the point where the two models are connected with each other.

Relational models were already employed with DEX in two real-world use-cases, in which they proved very useful for defining a decision model and evaluating alternatives on relational data: the reputational risk assessment of banks (Bohanec et al., 2013, 2014) and the appraisal of energy production technologies in Slovenia (B. Kontić, Bohanec, Trdin, et al., 2014; B. Kontić et al., 2016). Both are presented in Chapters 8, and 9, respectively.

4.4.1 Extension Formalization

To formalize this extension, we need to introduce three new entities and one type of aggregation function. The three new entities are: (1) *relational alternative*, (2) *relational model* and (3) *relational aggregated attribute*. The new aggregation function is called *relational aggregation function* and is used only by the relational aggregated attributes.

Given some alternative $a \in A$, the term *relational alternatives* denotes all entities $ra \in RA$ that are in a many-to-one relation with a. For instance, $a \in A$ may be a company and $ra \in RA$ one of its departments, assuming a one-to-many relation $department : A \mapsto RA$. Since RM, as any other model, contains input attributes, it also holds that

$$ra \in inputSpace_{RM}. \tag{4.23}$$

The purpose of the *relational model* RM is to evaluate relational alternatives RA. RM is modeled in the same way as any other DEX model — the only difference is in its purpose in the evaluation procedure. It evaluates relational alternatives RA and provides inputs to the main model M, which evaluates alternatives A.

A *relational aggregated attribute* rx is a special type of attribute that provides a connection point between the main model M and the relational model RM. rx's scale is formalized in the same way as it is for the other attributes. The relational attribute rx is connected to its counterpart output attribute ox from RM — rx serves as an input to M. In other words, rx is a connecting point between M and RM: it receives output from RM (through attribute ox) and provides input to M. However note that ox can be any type of attribute — either root, other aggregated attribute, input attribute or even relational attribute.

As there is a one-to-many relation between A and RA, the aggregation at rx should aggregate all output values coming from RM that correspond to one alternative $(V_1, V_2, \ldots, V_n), n \geq 1, V_i \in ED_{ox}$, into a single input value $V \in ED_{rx}$ of M. Here, n is the number of relational alternatives $ra \in RA$ evaluated by RM that are part of a single alternative $a \in A$ evaluated by M.

Consequently, the *relational aggregation function* f_{rx} is special in the sense that it aggregates n values coming from ox rather than single values from descendant attributes. Formally:

$$V = f_{rx}(V_1, V_2, \ldots, V_n), \tag{4.24}$$

$$f_{rx} : (ED_{ox})^n \mapsto ED_{rx}. \tag{4.25}$$

In general, f_{rx} is any aggregation function defined for an arbitrary number of arguments — for example, min, max, sum, average, mean, count or percentile$_X$, $X \in [1, 99]$. The value of the function is computed with the corresponding function F_{rx} as per Equation (4.21). Note that the function is constrained to the types of ox's and rx's scales. For instance, sum cannot be applied to a qualitative scale of rx.

The visual general configuration of the relational model is presented in Figure 4.8. Model M is on the top, connected to the relational model RM via attribute ox in RM and rx in M. The corresponding functions are f_{ox} (used for regular aggregation of values) and f_{rx}, used for relational aggregation. The alternative currently evaluating is a and alternatives corresponding to a are $ra \in RA$. Values $(V_1, V_2, \ldots, V_n)$ represent the n evaluations of relational alternatives from RA. From these, value $V \in ED_{rx}$ is computed using f_{rx}.

Given this formalization, an arbitrary number of relational models is supported in some model M. Moreover, an arbitrary number of relational models can be nested inside each other. Therefore, decision problems with nested relational alternatives can be modeled as well.

With this formalization, the *evaluation* procedure presented in Algorithm 3.1 has to be changed, to support relational aggregation. The new evaluation procedure is presented in Algorithm 4.2. The main difference to the initial evaluation algorithm is the different

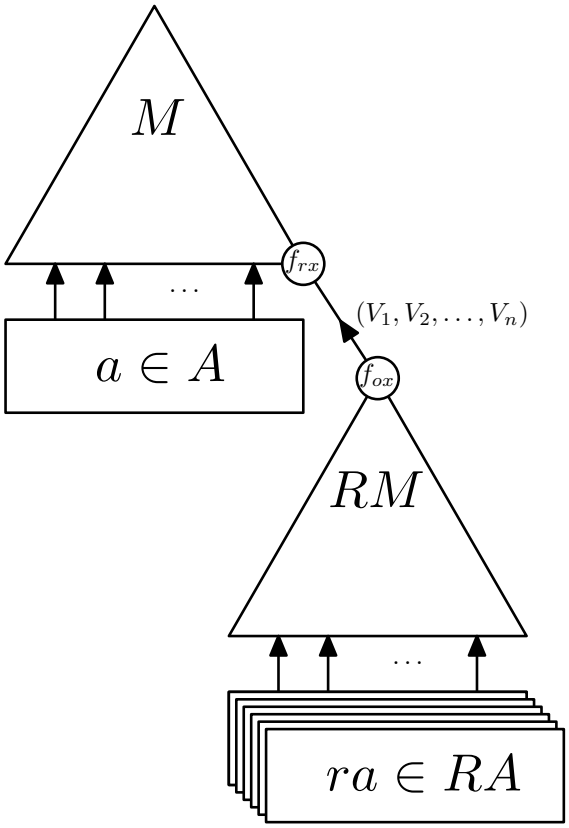

Figure 4.8: Configuration of connections between the main model and relational model. Here, M is the main model and RM the relational model, used for evaluation of relational alternatives $ra \in RA$ (corresponding to the alternative $a \in A$), using the aggregation function f_{rx}. The intermediate values produced by f_{ox} are $(V_1, V_2, \ldots, V_n)$.

handling of the input attributes (see lines 14–25). When the evaluation encounters an input attribute, which is also a relational attribute, it gets the corresponding relational model RM, corresponding relational alternatives RA and aggregation function F. It evaluates each relational alternative $ra \in RA$ against relational model RM, and adds the evaluation of the counterpart attribute ox into the array of evaluations. Finally, it computes the relational aggregation function F on the list of evaluations $relationalEval$, and adds the final evaluation to the alternative a_i.

Algorithm 4.2: Algorithm *evaluate* for evaluation of alternative a_i on model M, which also supports relational evaluation.

Input: Alternative a_i, Model M
Output: $(v_j, v_k, \ldots, v_l) \in outputSpace_M$

```
1  sort = topologicalSort(M.X);
2  S = M.S;
3  for j = 0; j < size(M.X); j++ do
4      x = sort[j];
5      if size(S(x))> 0 then
6          f_j = x.f;
7          values = [size(S(x))];
8          l = 0;
9          for Attribute x_k: S(x) do
10             values[l] = a_i[x_k];
11             l++;
12         end
13         a_i[x] = f_j(values);
14     else if size(S(x))== 0 and x is relational attribute then
15         RM = x.relationalModel;
16         RA = a_i.relationalAlternatives(RM);
17         F = x.relationalFunction;
18         ox = x.relationalConnection;
19         relationalEval = size(RA);
20         l=0;
21         for Relational Alternative: ra do
22             V = evaluate(ra, RM);
23             relationalEval[l] = V[ox];
24             l++;
25         end
26         a_i[x] = F(relationalEval);
27 end
28 output = [size(outputSpace_M)];
29 j = 0;
30 for Attribute x_l: outputSpace_M do
31     output[j] = a_i[x_l];
32     j++;
33 end
34 return output;
```

Chapter 5

Illustrative Example

This chapter continues development of the illustrative example, presented in Sections 3.1.1, 3.2.3, and 3.3.1. In the example, we firstly include three numeric attributes and a new aggregation function (as presented in Section 4.2). Then, we extend the input values of the alternatives to be value distributions (as presented in Section 4.3). Finally, we introduce an additional relational model with two attributes and three additional input attributes to the original models (as presented in Section 4.4). Correspondingly, we extend the three decision alternatives (apartments) and evaluate them with the extended model.

5.1 Numeric Attributes

We continue work on the illustrative example, which we evaluated in Section 3.3.1. We introduce numeric attributes in the example while keeping its hierarchical structure intact. First, we replace the scales of attributes *price*, *buying price* and *utilities* with their numeric counterparts (see Table 5.1). Due to these conversions, the aggregation functions of the modified attributes' parents (*price* and *apartment*) need to be replaced by functions of a different type. The remaining function for attribute *layout* remains the same (Function type $QQ \mapsto Q$).

Table 5.1: The model for assessment of apartments, with numeric attributes. On the left side, the structure of the attributes is presented, and on the right side, their corresponding scales are displayed. The scales are ordered from worst values (displayed in red) to best values (displayed in green). The numeric scales are also preferentially ordered, in a decreasing order.

Attribute	Scale
apartment	very bad, bad, ok, good, very good
price	$(0, \infty)$
buying price	$(0, \infty)$
utilities	$(0, \infty)$
location	bad, ok, good
layout	bad, ok, good
interior	bad, ok, good
exterior	bad, ok, good

Table 5.1 presents the structure of the developed model, with the respective scales of the attributes. The structure of the model did not change, however the scales for the *price*,

buying price and *utilities*. Given the meaning of the numeric attributes we assume that their numeric scales are all positive real numbers. Formally, the new scales are:

$$\begin{aligned} D_{\text{price}} &= (0, \infty), \\ D_{\text{buying price}} &= (0, \infty), \\ D_{\text{utilities}} &= (0, \infty). \end{aligned} \tag{5.1}$$

Note that all other scales remain the same, as described in Equation (3.11).

Further, to completely define the model, we need to define the two new aggregation functions: f_{price} and $f_{\text{apartment}}$.

The new function for *price* is of *Function type 3* ($NN \mapsto N$). Suppose that we pay off the apartment in 20 years, so our combined price will represent the buying price with 20 years of monthly utilities costs added:

$$f_{\text{price}}(\text{buying price}, \text{utilities}) = \text{buying price} + 12 \cdot 20 \cdot \text{utilities}. \tag{5.2}$$

Additionally, we need to define the aggregation function for *apartment*, which is of the *Compound type* $NQ \mapsto Q$. We shall compose it according to the schema $(NQ \mapsto N) \mapsto Q$ — that is: (1) first using the primitive *Function type 5*, and (2) discretizing the result. The output of (1) is an estimate of price paid (see Table 5.2). Because we wanted to include some preferential information in the table, we deducted constant values to favorable qualitative combinations (e.g. $-10\,000$ for the combination (*good*, *good*) in Table 5.2). For the unfavorable combinations, we added constant values (e.g. $+20\,000$ for the combination (*bad*, *bad*)).

Table 5.2: The intermediate aggregation function for apartment evaluation from price, location and layout. Every combination of location and layout input values (on the left) results in a function of price (right-most column).

location	*layout*	Φ(apartment)
bad	bad	price $+ 20\,000$
bad	ok	price $+ 15\,000$
bad	good	price $+ 5\,000$
ok	bad	$\frac{\text{price}}{1.1} + 7\,500$
ok	ok	$\frac{\text{price}}{1.1}$
ok	good	$\frac{\text{price}}{1.1} - 5\,000$
good	bad	$\frac{\text{price}}{1,1}$
good	ok	$\frac{\text{price}}{1.1}$
good	good	$\frac{\text{price}}{1.1} - 10\,000$

In step (2), we use the discretization of the achieved output value, from the step (1) of the function:

$$NQ_{\text{apartment}}(y) = \begin{cases} \text{very good} & \text{if } y \in [-10\,000, 55\,000] \\ \text{good} & \text{if } y \in (55\,000, 90\,000] \\ \text{ok} & \text{if } y \in (90\,000, 110\,000] \\ \text{bad} & \text{if } y \in (110\,000, 140\,000] \\ \text{very bad} & \text{if } y \in (140\,000, \infty) \end{cases}. \tag{5.3}$$

With this, the model is completely defined and ready for the evaluation of alternatives. Note that the evaluation order of attributes stays the same because the model structure did not change. The new numeric input values for the alternatives are presented in Table 5.3. The remaining qualitative input values remain unchanged. All evaluation results are given in Table 5.4.

Table 5.3: The three considered alternatives in the apartment evaluation, with numeric input values. For each of the model input attributes, the respective alternative has a value assigned.

Name	*buying price*	*utilities*	*location*	*interior*	*exterior*
Big	85,000	100	bad	bad	ok
Equipped	80,000	60	ok	ok	good
Nice	50,000	90	good	bad	bad

Let us illustrate the evaluation of the alternative *Big*. The other two alternatives follow the same procedure. At first, the price of alternative *Big* is obtained by computing $f_{\text{price}}(85\,000, 100)$, which is 109000. The value of *layout* is computed using the $QQ \mapsto Q$ function $f_{\text{layout}}(\text{bad}, \text{bad})$, which produces bad. Finally, the value of function for *apartment*, which is of *Compound type* $NQ \mapsto Q$, is computed according to schema $(NQ \mapsto N) \mapsto Q$. The first step selects the function price + 20 000, where price = 109 000, so the resulting value is 129000. Using $NQ_{\text{apartment}}$, this value discretizes to *bad* — the overall evaluation of apartment *Big*. All evaluations of the remaining alternatives are presented in Table 5.4.

Table 5.4: The three considered alternatives in the apartment evaluation with included numeric scales, and the evaluations of the respective aggregated attributes.

Name	*price*	*layout*	*apartment*
Big	109,000	bad	bad
Equipped	94,400	ok	good
Nice	71,600	bad	good

Let us compare these results with the final evaluations acquired in Section 3.3.1. Because of the addition of numeric attributes, values of *price* attribute changed — consequently final evaluations of apartments changed. The most notable difference is the change of the final evaluation of alternative *Equipped*, which increased from *bad* to *good*. Due to added numerical attributes and changed aggregation functions, the final price is mapped to *good* value. Consequently, the apartments *Equipped* and *Nice* are both evaluated as *good*. Now, *Equipped* differs from *Nice* in terms of a better *layout*, but with a worse *price*. Both have the same final evaluation — *good*. The final evaluation of apartment *Big* increased from *very bad* to *bad* because of the same reasons.

We can see that even though the developed model is more complex, the introduction of numeric attributes brought additional modeling possibilities to our illustrative example. The inclusion of numeric values allowed for a more natural (monetary) interpretation of price concepts in comparison to describing them with qualitative values.

5.2 Value Distributions

We continue the illustrative example from the point at which we left it in the previous section. The model now contains numeric values, and hence the alternatives are also defined using numeric values. Hereafter, we shall include value distributions at alternatives' input values. The aggregation functions are left unchanged. The type of aggregation is probabilistic.

We define probability value distributions of alternatives according to the following observations:

- Alternative *Big* does not actually have a fixed utilities bill of 100 — rather, the value is uniformly distributed through the year in the amount from 80 to 120. The same applies to alternative *Nice*, but the value is distributed between 60 and 80. However, the utilities costs for apartment *Equipped* are still fixed at 60.
- We do not know the quality of the neighbourhood of alternative *Equipped*. For this reason we previously decided to assign the middle value. Now we are able to specify that the value is actually *undetermined* or *unknown*, assigning an interval of all possible values.
- The equipment of apartment *Nice* was previously determined by rule of thumb, and we felt that specifying a single value was too limited. Instead we now specify the situation with a distribution.
- We were not sure before about the *exterior* of the apartments *Equipped* and *Nice*, so we decided with a rule of thumb. Now the two input values are distributions, showing the actual state of the apartment's exterior.

All observations are quantified in Table 5.5, with all remaining input values.

Table 5.5: The three considered alternatives in the apartment evaluation, with numeric input values and probabilistic value distribution. For each of the model input attributes, the respective alternative has a value assigned.

Name	*buying price*	*utilities*	*location*	*interior*	*exterior*
Big	85,000	$U(80, 120)$	bad	bad	ok
Equipped	80,000	60	[bad, good]	ok	{bad/0.1, good/0.9}
Nice	50,000	$U(60, 80)$	good	{bad/0.6, good/0.4}	{bad/0.9, good/0.1}

Because the model aggregation functions and scales of attributes did not change from the previous example, we can immediately evaluate the alternatives. The order of aggregation function evaluations stays the same as before — the model structure stayed the same. As in previous examples, we will show the evaluation of the first alternative *Big* and present the results of the evaluations of the other two alternatives. All evaluations of alternatives are summarized in Table 5.6.

Table 5.6: The three considered alternatives in the apartment evaluation with value distributions at inputs, and the evaluations of the respective aggregated attributes.

Name	*price*	*layout*	*apartment*
Big	{104 212/0.01, 111 870.4/0.01, ..., 106 600/0.01}	bad	bad
Equipped	94,400	{bad/0.1, ok/0.9}	{bad/0.03, ok/0.33, good/0.63}
Nice	{64 431/0.01, 68 204/0.01, ..., 67 184/0.01}	{bad/0.54, ok/0.42, good/0.04}	{good/0.96, very good/0.04}

First, *Big*'s price is computed using f_{price} with argument values of 85000 and $U(80, 120)$. The values are interpreted as value distributions: 85000 is $\{85\,000/1\}$ and $U(80, 120)$ is sampled m times (we choose $m = 100$), obtaining the value distribution $\{80.05/0.01, 111.96/0.01, \ldots, 90/0.01\}$. For all 100 combinations of values in buying price ($\{85\,000/1\}$) distribution and utilities distribution ($\{80.05/0.01, 111.96/0.01, \ldots, 90/0.01\}$), the function f_{price} is evaluated during computation of F_{price}. The final produced evaluation for price is $\{104\,212/0.01, 111\,870.4/0.01, \ldots, 106\,600/0.01\}$.

The value for *layout* is evaluated to *bad* as before. The final evaluation for apartment *Big* is acquired with function $F_{\text{apartment}}$. The values supplied to the function are $\{104\,212/0.01, 111\,870.4/0.01, \ldots, 106\,600/0.01\}$, *bad* and *bad*. The first value is already a value distribution, and from the other two values, value distributions $\{\text{bad}/1.0\}$ are made. For each combination of values, $f_{\text{apartment}}$ is queried for a value. Note that the probability for each value combination is 0.01. Because the only possible values for location and layout are *bad*, function price $+\,20\,000$ from space Φ(apartment) is always invoked. The first value from the distribution gives 124212, which is later mapped to value *bad* by $NQ_{\text{apartment}}$. This is done for all crisp values inside the value distribution of *price*. Disregarding the actual value of *price*, $NQ_{\text{apartment}}$ always produces *bad*, which in turn, after simplification, produces the final value evaluation of *bad*. The two other alternatives are similarly evaluated. The results are presented in Table 5.6.

The final evaluations now give us an additional insight regarding our alternatives. Even though we know that utilities costs will be variable for alternative *Big*, the final evaluation of this apartment is still *bad*. On the other hand, we can now better compare apartments *Equipped* and *Nice*. With only numeric attributes included in the model, we were not able to distinguish between the apartments in terms of the final evaluation. Now we can see that apartment *Equipped* is mostly good (with probability 0.6), but there is the probability of 0.37 that the evaluation is a *ok*, or even *bad* (with probability 0.03). For the apartment *Nice*, the evaluation is strongly *good* (probability 0.96) with a small chance (probability 0.04) that the apartment is even *very good*. It seems that apartment *Nice* is better than apartment *Equipped*. We can confirm that with stochastic dominance (see Equation (4.22)):

For that, we check for each value $v \in D_{\text{apartment}}$ that the probability of the value v or better in the evaluation of *Nice* (V_{Nice}) is greater or equal than the probability of value v or better in the evaluation of *Equipped* (V_{Equipped}):

$$\begin{aligned}
1 = P[V_{\text{Nice}} \geq \text{very bad}] &\geq P[V_{\text{Equipped}} \geq \text{very bad}] = 1,\\
1 = P[V_{\text{Nice}} \geq \text{bad}] &\geq P[V_{\text{Equipped}} \geq \text{bad}] = 1,\\
1 = P[V_{\text{Nice}} \geq \text{ok}] &\geq P[V_{\text{Equipped}} \geq \text{ok}] = 0.97,\\
1 = P[V_{\text{Nice}} \geq \text{good}] &\geq P[V_{\text{Equipped}} \geq \text{good}] = 0.6,\\
0.96 = P[V_{\text{Nice}} \geq \text{very good}] &\geq P[V_{\text{Equipped}} \geq \text{very good}] = 0.
\end{aligned} \tag{5.4}$$

Additionally, for the apartment *Nice* to stochastically dominate apartment *Equipped*, we need to find a value $v' \in D_{\text{apartment}}$, for which the probability of producing value at least v' in V_{Nice} is strictly greater than probability of producing at least value v' in V_{Equipped}. There are three such values: *ok*, *good* and *very good*.

Consequently, *Nice* stochastically dominates *Equipped*, and can be considered a better choice. In the same way, it can be shown that both *Nice* and *Equipped* dominate *Big*.

5.3 Relational Model

To show the relational models in practice, we will once again extend the previously developed illustrative example, taking into account that each apartment consists of multiple

rooms. Each room has some features worth evaluating, such as size and equipment. Thus, in addition to the main model for apartments, we introduce a relational model for the evaluation of rooms. The relational model contains two attributes: (1) *size*, a numeric attribute giving the absolute size of the room and (2) *equipment*, a qualitative attribute giving the amount and quality of equipment in the room.

Due to the nature of this design, three relational aggregated attributes are introduced in the apartment model as inputs to the *interior* attribute: *size*, *rooms* and *equipment* of the apartment (representing size of the whole apartment, number of rooms and the equipment in the apartment). The respective scales of the added relational attributes are: positive real numbers, positive integer numbers, and *no*, *some*, *yes*. The new structure for the apartment model, together with the corresponding scales is presented in Table 5.7. The table shows the attributes in black (regular attributes) and blue color (relational attributes). The descendant attributes from attributes written in blue are the counterpart relational input attributes.

Table 5.7: The model for assessment of apartments, with relational attributes added. On the left side, the structure of the attributes is presented, where attributes in black are regular attributes and attributes in blue color are relational attributes. Attributes descendants of blue attributes are the relational counterpart attributes. On the right side, the corresponding scales are displayed. The scales are ordered from worst values (displayed in red) to best values (displayed in green). The numeric scales for price attributes are also preferentially ordered, in a decreasing order, while the size's and room's scale are ordered in an increasing order.

Attribute	**Scale**
apartment	very bad, bad, ok, good, very good
price	$(0, \infty)$
buying price	$(0, \infty)$
utilities	$(0, \infty)$
location	bad, ok, good
layout	bad, ok, good
interior	bad, ok, good
size	$(0, \infty)$
size	$(0, \infty)$
equipment	no, some, yes
equipment	no, some, yes
rooms	$[1 \ldots \infty)$
equipment	no, some, yes
exterior	bad, ok, good

For the *apartment* model to be complete, we need to specify the aggregation function for the *interior* aggregated attribute (f_{interior}). The function maps from two numeric inputs (*size* and *rooms*) and qualitative *equipment* to qualitative values of *interior*. This function, similarly to function $f_{\text{apartment}}$, is of *Compound type* $NQ \mapsto Q$. We shall compose it according to the schema $(NQ \mapsto N) \mapsto Q$ — that is: (1) first using the primitive *Function type 5* and (2) discretizing the result. The output of (1) is an estimate of each room size (see Table 5.8). Because we wanted to include some preferential information in the table, we added constant values to favorable qualitative combinations (e.g. +20 for the *equipment* being *yes* in Table 5.8). For the unfavorable *no*, we disregarded the numeric inputs and assigned the value 0.

In step (2), we use the discretization of the achieved output value, from the (1) step of

Table 5.8: The intermediate aggregation function for interior evaluation from size, equipment and rooms. For each value of equipment (on the left) a resulting function of size and rooms is produced (right-most column).

equipment	Φ(interior)
no	0
some	$\frac{\text{size}}{\text{rooms}}$
yes	$\frac{\text{size}}{\text{rooms}} + 20$

the function:

$$NQ_{\text{interior}}(y) = \begin{cases} \text{bad} & \text{if } y \in [0, 25] \\ \text{ok} & \text{if } y \in (25, 35] \\ \text{good} & \text{if } y \in (35, \infty) \end{cases}. \tag{5.5}$$

Table 5.9 presents the relational model, for *room* assessment. In this case, the model is simply (linearly) structured. Both attributes are input and also output attributes of the model. Hence, the model for *room* assessment does not need any aggregation function and is complete.

Table 5.9: The model for assessment of rooms. On the left side, the structure of the attributes is presented, while on the right side, the corresponding scales are displayed. The qualitative scale is ordered from worst values (displayed in red) to best values (displayed in green). The numeric scale for size attribute is also preferentially ordered, in an increasing order.

Attribute	**Scale**
size.....	$(0, \infty)$
equipment	no, some, yes

The actual configuration between the main *apartment* model and the relational *room* model is presented in Figure 5.1. Multiple evaluations of rooms (attributes ox) influence relational attributes (rx). Concretely, *size* attribute (in model *room*) influences relational attribute *size* in *apartment* model. Attribute *equipment* from model *room* influences both *equipment* and *rooms* in *apartment* model.

For the models to be usable for evaluation of apartments, we need to specify the relational aggregation functions for the relational attributes. The values are computed as follows: (1) *size*'s value is computed by summing the values of *size* attribute from the room relational model, (2) *equipment* transforms the room equipment values with uniform weighting and (3) *rooms* counts the number of rooms by counting the number of *equipment*'s values that come from the relational model. The functions are formalized as follows:

$$\begin{aligned} f_{\text{size}}(v_1, v_2, \ldots, v_n) &= \sum_{i=1}^{n} v_i, \\ f_{\text{equipment}}(v_1, v_2, \ldots, v_n) &= \frac{1}{n} \sum_{i=1}^{n} v_i \\ f_{\text{rooms}}(v_1, v_2, \ldots, v_n) &= count(v_1, v_2, \ldots, v_n) = n. \end{aligned} \tag{5.6}$$

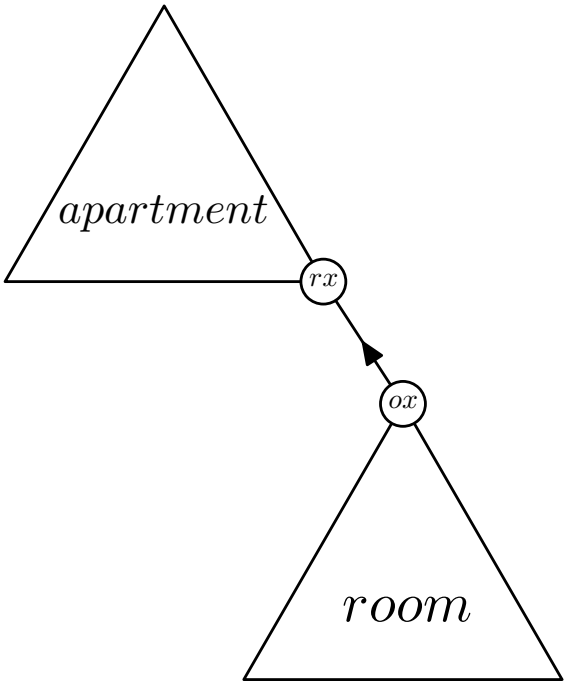

Figure 5.1: The configuration of main model for apartment assessment with the relationally connected model for room assessment. Multiple rooms influence evaluation of a single apartment.

To use the new model, we have to define input values for rooms, and remove the input values for the *interior* attribute. Other input values remain unchanged, from the usage in previous section. The input values are presented in Table 5.10.

Table 5.10: The three considered alternatives in the apartment evaluation, with numeric input values, probabilistic value distributions and with input values for relational alternatives (rooms). The input values for rooms are given to the right of the respective parent apartment alternative.

Name	*buying price*	*utilities*	*location*	*exterior*	Room alternatives	
					size	*equipment*
Big	85,000	$U(80, 120)$	bad	ok	40	no
					25	no
					15	{no/0.9, some/0.1}
Equipped	80,000	60	[bad, good]	{bad/0.1, good/0.9}	25	yes
					15	{some/0.1, yes/0.9}
Nice	50,000	$U(60, 80)$	good	{bad/0.9, good/0.1}	30	{some/0.6, yes/0.4}

The evaluation procedure is almost the same as before. We can follow a similar topological sorting. The only difference is that the *interior* attribute is now aggregated attribute and must be included in the sorting:

$$(\text{price}, \text{interior}, \text{layout}, \text{apartment}). \tag{5.7}$$

The value for the *price* is computed in the same way as before. However, some inputs to the sub-tree of *interior* are now obtained from the relational model and may

differ from previous calculations (recall, previously *interior* was an input attribute). The main difference is in the relational aggregation that involves functions f_{size}, $f_{\text{equipment}}$ and f_{rooms}. The aggregation function f_{size} first evaluates the *size* of three rooms of the *Big* apartment, which gives values of $\{40, 25, 15\}$. Function produces the sum, which is 80. Similarly, the value for *equipment* is computed. The evaluation on rooms model gives $\{\text{no}, \text{no}, \{\text{no}/0.9, \text{some}/0.1\}\}$. The function $f_{\text{equipment}}$ weighs each value with $\frac{1}{3}$, and computes the final distribution of $\{\text{no}/0.97, \text{some}/0.03\}$. This particular distribution cannot be simplified and is left as is. The computation of *rooms* follows the same principle as the computation for *equipment*, except that $f_{\text{equipment}}$ counts the number of elements rather than creates a weighted sum of values — the function produces value 3. These values now enter the calculations as ordinary inputs of the apartment model, giving the results as shown in Table 5.11.

Table 5.11: The three considered alternatives in the apartment evaluation with value distributions at inputs, also considering the relational evaluations.

Name	*price*	*size*	*equipment*	*rooms*	*interior*	*layout*	*apartment*
Big	{104 212/0.01, 111 870.4/0.01, ..., 106 600/0.01}	80	{no/0.97, some/0.03}	3	{bad/0.97, ok/0.03}	{bad/0.97, ok/0.03}	bad
Equipped	94,400	40	{some/0.05, yes/0.95}	2	{bad/0.05, good/0.95}	{bad/0.005, ok/0.14, good/0.855}	{bad/0.002, ok/0.33, good/0.665}
Nice	{64 431/0.01, 68 204/0.01, ..., 67 184/0.01}	30	{some/0.6, yes/0.4}	1	{ok/0.6, good/0.4}	{bad/0.54, ok/0.42, good/0.04}	{good/0.96, very good/0.04}

The final evaluation of alternative *Big* did not change due to the relational aggregation functions — the computed value is the same as before. The same holds for the *Nice* apartment. However, the new evaluation of *Equipped* is different. Previously, the value of *interior* was *ok*. Now, as we explicitly considered two rooms, one of which is small and incompletely equipped, the aggregation of the rooms gives the value distribution $\{\text{some}/0.05, \text{yes}/0.95\}$. Thus, the values *some* and *yes* are additionally propagated through the aggregation, resulting in a new evaluation of $\{\text{bad}/0.002, \text{ok}/0.33, \text{good}/0.665\}$. Similarly, as in previous section, we can show that evaluation of apartment *Nice* stochastically dominates evaluation of apartment *Equipped*. Furthermore, the *Equipped*'s evaluation dominates the *Big*'s evaluation.

Chapter 6

DEXx Implementation

In the course of the thesis we developed an implementation of DEXx method as a library. The implementation provides a better and more powerful ground structure for decision making with DEX method. The developed library provides an access to the concepts and algorithms for extensions described in this thesis, to facilitate usage in different applications. These applications include usage in various programs in Java, standard shell programs, other developed computer programs, web sites supporting Java programming language and usage in web-services.

The DEXx library is a new implementation of DEX method, supporting the five extensions considered in this thesis. Specific contributions and advances are:

- Implementation of five extensions presented in this thesis. All the extensions can be concurrently used *together* in a decision making scenario.
- Implemented using one of the most popular programming languages — Java, which allows for wider code readability.
- Programming code can be ported to most popular operating systems and computer architectures.
- The library is implemented with important Java-based concepts, such as generics, object-oriented design and polymorphism.
- The compiled library and complete source files are available online, and are managed using *mercurial* version control.
- Implementation of specific functions on models, which facilitate further viewing and analysis of the developed models and evaluations (see Section 6.10).

This chapter presents the goals and the purpose of the implementations, details about software architecture used, hierarchy of classes in Java, and specific implementational details of most used objects in DEXx library. The chapter is concluded with presentation of important methods, used on models.

6.1 Goals and Purpose of Implementation

The main purpose of the implementation of decision support method DEXx is to facilitate and simplify the development of decision models in complex situations, which were hard or even impossible to address previously. The implementation is able to cope with the ever

increasing amounts and complexity in data, and handling this data in a timely fashion using the developed models.

The implementation provides a better and more powerful ground structure for decision making with DEX methodology. The new generation platform provides an easier access to the developed library to facilitate usage in different applications. The applications vary from Java applications to web-service implementation.

The goal of the implementation is to develop the existing DEX methodology in Java programming language. The implementation is backward compatible with the existing DEXi software package and is also transferable among many computer platforms. On top of the implementation of the DEX methodology, all five proposed extensions, presented in this thesis, are implemented and tested. Along with the extensions, all other needed functions and methods are developed, to facilitate the addition of a graphical user interface, in order for the application to be a stand-alone application with a user interface.

6.2 Software Architecture

As mentioned in the previous Section, for the purpose of the implementation we selected Java programming language. There were however five programming languages considered for the implementation: C++, C#, Java, Pascal and Python. These five languages were selected based on our knowledge and expertise in them. Pascal language also has a benefit, because DEXi (Bohanec, 2015b) is implemented with it. Each of the languages has its benefits and weaknesses:

C++ is a programming language designed by *Bell Labs*. It is a general-purpose imperative programming language, which is also object-oriented. Typical C++ applications are developed for low resource usage in embedded systems and in large systems where performance and efficiency are a need. It is a compiled language, which cannot be easily transferred between different computer architectures — the usage in different computer architectures requires a new compilation of the code and/or even the used libraries. The language supports many novel features and is being constantly developed. On the negative note, for the efficiency of the program, the usage of the language requires extensive memory management.

C# was initially developed by *Microsoft*, during their *.NET* initiative. It is a general purpose multi-paradigm language, supporting imperative, declarative, functional, generic, object-oriented and component-oriented programming paradigms. C# is a compiled language, which means that the language is compiled into machine code and is inherently fast. However, the down side is that for usage on different systems the program must be recompiled. Also, because the language was developed by Microsoft, it is fully functional on Windows based platforms — if the program uses .NET libraries. There is however no standard compiler for other platforms, which might prove difficult to handle in the future. The language provides some low-level features, such as memory management. However, the support is not as extensive as the one provided by C++.

Java is a programming language originally developed by *Sun Microsystems*. It is a concurrent, class-based and object-oriented. The implementation of the language allows the application developers to "write once, run anywhere". This means that the compiled Java code can run on any platform that supports Java, without need for recompilation. The Java code is typically compiled into *bytecode* (intermediate code similar to machine code), which is then run on any Java virtual machine, regardless of the

computer architecture. The language has a large standard library and many more other libraries that can be easily incorporated into applications. Java programming language however lacks extensive low-level facilities, such as memory management, which is available in C++.

Pascal is a historically influential imperative and procedural programming language, initially developed by Niklaus Wirth. A derivative of language (Object Pascal) was later designed for object-oriented design. Pascal is a compiled language, and compilers are available for all mayor computer platforms, however each compiler provides different libraries and additional constructs to the language. The most notable compiler is one provided by *Delphi*. Pascal provides facilities for memory management.

Python is an interpreted programming language initially developed by *Centrum Wiskunde & Informatica*. It is a high level programming language supporting object-oriented, imperative and functional programming designs. The code is very portable between different computer architectures, which is mainly due to the fact that the language is interpreted — the code can be executed on any architecture that has a Python interpreter. The main design philosophy of Python is to emphasize on code readability. The language features design concepts, which allows for fewer lines of code in comparison with implementations in Java or C++. It supports dynamic types, which eliminates the need for declaration of variables on one hand, however it becomes confusing in large projects. The most notable downside of Python against Java and C++ is its speed, which is quite slower, because the language is not compiled to machine code, rather it is interpreted by an interpreter.

Considering the five options for the selected programming language, we selected Java in the end. The selection was based on the facts that Java binaries are easily portable between different computer architectures and its binaries can also be used in web-based applications via *Java Applets*. Java also natively features automatic memory management and garbage collection, which in the end requires less work for the programmer. Another crucial feature which Java supports very well is the class-based and object-oriented design. Later in Section 6.3 we present the developed class hierarchy for main classes, interfaces and exceptions used in the implementation. The object-oriented design allows for four important concepts: information hiding between connected classes, inheritance of properties of sub-classes, polymorphism and interfacing. Polymorphism in the context of computer programming means the ability to replace an *object* with a *sub-object* from a sub-class in various places, i.e. in function calls. Interfacing in object-oriented design is an ability to define functions or methods (as their signatures) without actually implementing them. These four features are most useful in the particular context and are hence actively used in the implementation of the DEXx library.

The code repository is under version control with *mercurial*. The repository of the implemented library is publicly available on https://bitbucket.org/nejctrdin/dexx/ (Trdin & Bohanec, 2015). The web page gives an introduction to the library and presents a small working example of car evaluation. In the example, a DEXx model is constructed, with three attributes with qualitative scales and one aggregation function (*Function type 1*). Two car alternatives are defined and evaluated with the model — one with distribution as an input value. More complex usages of the library can be found under src/test/java/examples/FunctionExamples.java.

6.3 Class Hierarchy

In Java, the programming is inherently object-oriented. For that matter, we developed a class hierarchy that would hold all algorithms and data structures for the modeling and evaluation purposes. The hierarchy of classes is presented in Table 6.1. All classes are sub-classes of main Java class *Object*. The main class in the hierarchy is the *QObject* class, from which all other important classes inherit. QObject is an abstract class, and therefore cannot be instantiated. This is an empty class and is present for distinguishing the DEXx classes from other Java classes. QObject class has four direct descendant classes which are split according to their purpose:

- *QAlternative* is a class whose instances are alternatives in the DEXx method. The instances of the class hold the input values of the concrete alternatives and all their evaluations, together with the *QModel* instance the alternative belongs to. QAlternative also has one descendant class, *QRelationalAlternative*. The class is used for storing relational alternatives. Along with all inherited properties of the QAlternative, it also holds a reference to the instance of QAlternative the particular relational alternative belongs to.
- *QNode* is an abstract class which differs from other classes by the fact that all descendants of the class have *some inputs* and *some outputs*. All descendants implement the Nameable interface. Here inputs are defined in a vague way — they may come from any instance of the descendants of QNode class. The QNode class has one descendant instantiationable class, *QModule* for representing modules. The purpose of the module is to group together various different modules, models or variables (attributes). The module's main property is that it has at least one input and at least one output. Furthermore, QModule has two descendant classes:
  - *QModel* is the main class whose instances hold the model in DEXx. The model holds references to all used variables and alternatives. It implements the Owner interface. It also guides the evaluation process, while evaluating alternatives. The class has one descendant, *QRelationalModel* for storing relational models. Along with the inherited attributes from QModel, the QRelationalModel stores a reference to its parent model (or relational model).
  - *QVariable* is a class which instantiates to an attribute in the model. The instances of the class hold a reference to the scale (QScale) and a reference to an aggregation function (QAggregationFunction), if the attribute is an aggregated attribute. Additionally, it has a reference to the parent attributes and the descendant attributes (if they exist). It implements the Owner and the Subordinate interfaces. The variable is the owner of the aggregation functions, and a subordinate to the model. Furthermore, it implements the User interface, for using the shared aggregation functions and/or scales. Its sub-classed *QRelationalVariable* additionally holds a reference to the relational model and respective counterpart relational inputs. Also, it holds a reference to relational aggregation function (QRelationalAggregationFunction), instead of QAggregationFunction.

QSharedObject is the third descendant of the QObject class. It is an abstract class, which gives the user a possibility to share a same instance of the class in multiple contexts. Hence, all descendants implement the Shared interface. There are two uses for the sharing:

- *QAggregationFunction* is an abstract class that receives a determined number of input values from input domains and produces an output from a specific domain. It has eight descendant classes:

  - *QDebugFunction* is a simple function implementation, which is used for debugging purposes. It merely returns a string concatenation of input values.
  - *QGeneralAggregationFunction* is an abstract function, which is not used in DEXx directly. Rather it is a class which can be extended by the user application, and enables the user to implement a function outside of DEXx framework. For example, such function can call an external web-service or query a database.
  - *QQualitativeToQualitativeFunction* is a class representing functions of *type 1*. Internally, it models the function as a look-up table of qualitative input values to a qualitative output value.
  - *QQualitativeToQuantitativeFunction* is a class representing instances of the *type 2* functions. It is modeled as a look-up table of qualitative input values to a quantitative value.
  - *QQuantitativeToQualitativeFunction* is used for functions of *type 4*. It holds a mapping from numeric intervals to a qualitative output value.
  - *QQuantitativeToQuantitativeFunction* is used for functions of *type 3*. Internally, the function is modeled purely as a mathematical function, which maps from numeric values to a numeric output value.
  - *QQuantitativeQualitativeToQuantitativeFunction* is used for instantiation of *type 5* functions. Internally it is composed of a look-up table, mapping from qualitative input values to a function of other numeric inputs. The function then maps from numeric inputs to a numeric output.
  - *QRelationalAggregationFunction* is a type of aggregation function used in relational attributes. Internally it holds a reference to an operation on n values — for instance sum, weighted sum or product.

- *QScale* is used for the purpose of determining the type of the attributes scale. Additionally to the type of values it holds a reference to the upper and lower limit of the accepted values (for the numeric scales) or a list of all possible values (for qualitative scales). The instance also holds a property, if the scale is preferentially ordered.

- *QValue* is the fourth descendant class of QObject. It is an abstract class, whose sub-classed classes hold a specific value used in aggregation functions and in scales. It has two sub-classes:

  - *QComplexValueType* is an abstract class used for representing the values of the extended domain. Internally, each of the sub-classes holds values from QSimpleValueType. *QFuzzyValue* additionally to values holds the membership function values. For a difference, *QDistributionValue*'s membership values are called probabilities and sum up to 1. *QIntervalValue*'s speciality is that the held values are representing the lower and the higher limits of the interval from a specific scale, whereas the *QSetValues*' are arbitrary values from some scale.
  - *QSimpleValueType* is an abstract class. Instances of its sub-classes are used in the scales and internal representation of inputs in aggregation functions. Each instance holds only one value of real type *QDoubleValue*, integer type *QIntegerType* or a qualitative type *QQualitativeValue*.

Additionally to *QObject*, there are four important classes used in the running of the library which are sub-classed from Java's Object:

- *GlobalProperties* is a class which can load different options for running the library. The most notable option is the writing of the log file, location of the file and its severity level.

- *Log* is a class to which all above mentioned classes report their behavior and errors. It then guides the writing to the log file according to the specified preferences.

- *ObjectRepository* is a class that tracks all created objects. Its main purpose is for debugging of the library.

- *OperationRegistry* is a class that records the interactions between the objects described above. Mainly it registers the operations (from interface *Operation*) that subordinates can perform in the owner's context. For example, if a variable can change its scale in the context of the model.

Table 6.1: The hierarchy of main Java classes in the implemented library. These classes are the building blocks of each model developed with DEXx.

Class

```
Object
├── QObject
│   ├── QAlternative
│   │   └── QRelationalAlternative
│   ├── QNode
│   │   └── QModule
│   │       ├── QModel
│   │       │   └── QRelationalModel
│   │       └── QVariable
│   │           └── QRelationalVariable
│   ├── QSharedObject
│   │       ├── QAggregationFunction
│   │       │   ├── QDebugFunction
│   │       │   ├── QGeneralAggregationFunction
│   │       │   ├── QQualitativeToQualitativeFunction
│   │       │   ├── QQualitativeToQuantitativeFunction
│   │       │   ├── QQuantitativeToQualitativeFunction
│   │       │   ├── QQuantitativeToQuantitativeFunction
│   │       │   ├── QQuantitativeQualitativeToQuantitativeFunction
│   │       │   └── QRelationalAggregationFunction
│   │       └── QScale
│   └── QValue
│       ├── QComplexValueType
│       │   ├── QFuzzyValue
│       │   │   └── QDistributionValue
│       │   ├── QIntervalValue
│       │   └── QSetValue
│       └── QSimpleValueType
│           ├── QDoubleValue
│           ├── QIntegerValue
│           └── QQualitativeValue
├── GlobalProperties
├── Log
├── ObjectRepository
└── OperationRegistry
```

Java also supports *interfaces*, which enables the instances of classes to have certain properties or conform to some standards. For that purpose, we also used the interfaces in the implementation of DEXx. The interface hierarchy is presented in Table 6.2. There are six interfaces, which all descend from the Java's Object class:

- *Nameable* is an interface which defines that objects implementing the interface should have a name, and return the name of the particular object with a method. The interface is used in all sub-classes of *QNode*.
- *Operation* is an interface which the operation *enums* of *Owners* must implement. This means that each owner enables its respective registered subordinates to perform some operations in the context of the owner. If the operation is not registered, then by definition it is prohibited.
- *Owner* is an interface which is assigned to sub-classes of QNode. These instances can be owners of their context, and allow or disallow changing of the models by the registered subordinates.
- *Shared* is an interface specified for using the sub-classes of the QSharedObject in a shared context. This enables for attributes to share a same scale and/or a same aggregation function, if it is applicable. The users of a shared object must implement the User interface.
- *Subordinate* is an interface used for specifying the objects, which are able to change the context of objects implementing the Owner interface. This interface is assigned to all descendants of QNode.
- *User* is an interface assigned to the QVariable class and is used to define the users of the shared objects. Only the attributes can be owners of the shared resource such as a scale or an aggregation function.

Table 6.2: The interfaces used in the Java library, used for assigning specific properties to the main classes of DEXx.

Class
Object
├─ Nameable
├─ Operation
├─ Owner
├─ Shared
├─ Subordinate
└─ User

For the reporting of errors occurring during the building of models and evaluating the alternatives in DEXx we implemented 11 exceptions that are arranged into a hierarchy of classes. The exceptions implemented are presented in Table 6.3. All exceptions are sub-classed from the Java's Exception class. We decided to sub-class it from Exception, because all its descendant exceptions can be *checked* — application can detect them and possibly recover from them. Whereas the Java's Error class does not enable this option — it specifies errors that the application cannot recover from. The main exception used in the library is *LibraryException* which is thrown if an implementational library error occurs. There is only one sub-class from it, the *GeneralModelException*, which is thrown when a

general exception occurs and it does not fit in any other more specific exception class. This exception is also used when there are problems with the main model — the arrangement of attributes forms a cycle or when an attribute performs an illegal operation. There are nine exceptions that are descendants from GeneralModelException:

- *AggregationFunctionException* is an exception, which is thrown when an error occurs while evaluating or defining a non-relational aggregation function. Its descendant *RelationalAggregationFunctionException* is thrown when problems in evaluation of relational aggregation function occur.

- *AlternativeException* is used when there is a problem with defining input values to the alternatives or when the evaluation value does not fit into the scale of the attribute. Similarly, the *RelationalAlternativeException* is thrown when problems are encountered in relational alternatives.

- *RelationalModelException* is thrown when any problem in a relational model occurs. For example, when a relational model must be evaluated, but does not have any relational alternatives.

- *ScaleException* is used for reporting exceptions during the construction of scales. For example, if the upper limit in a numeric attributes is lower than the lower limit, or a qualitative scale has two identical values.

- *ValueException* is thrown for all value related problems. Examples are when the probabilities in the distribution do not sum up to 1 or the interval value has the lower value preferentially higher than the higher value of the interval.

- *VariableException* is used for any failed checks and problems encountered in the attributes. The most frequent exception is reported if the aggregation function definition does not meet the needs of attribute — if the multiplicity and types of arguments or output does not match the aggregation function. Similarly, a *RelationalVariableException* is thrown for problems in relational attributes.

Table 6.3: The hierarchy of exceptions used in the implemented library. The exceptions are used for reporting errors during construction and evaluation of models developed with DEXx.

Class
Exception
├─ LibraryException
│ └─ GeneralModelException
│ ├─ AggregationFunctionException
│ │ └─ RelationalAggregationFunctionException
│ ├─ AlternativeException
│ │ └─ RelationalAlternativeException
│ ├─ RelationalModelException
│ ├─ ScaleException
│ ├─ ValueException
│ └─ VariableException
│ └─ RelationalVariableException

6.4 Attributes

Attributes are the most basic and the most important building block in the DEXx implementation. Attributes are implemented in class *QVariable*. They keep references to their domain's and aggregation functions (if the attributes themselves are aggregated). Furthermore, they also keep references of their direct parent attributes and direct child attributes.

During the initial modeling phase of MCDA, the algorithms for checking if the model is consistent need to be in place. Namely, the algorithm for cycle detection must check if the model structure contains a cycle on each addition of an attribute to the model. For that purpose we implemented the *topological sort* which returns a topological sort of attributes, if the model structure does not form a cycle. Otherwise, the algorithm returns an error. Topological sort enables the library to perform atomic operations on attributes while progressing through the sort, so that it guarantees for every attribute that lower-laying attribute's atomic operation has already been performed. This fact is most important in evaluation of models — where all lower-laying attribute values must be computed, before the current function is evaluated. The algorithm's pseudo code is presented in Algorithm 6.1.

Algorithm 6.1: Algorithm for topological sorting of the model structure in DEXx. The algorithm produces a sort of attributes, if the model structure does not have a cycle, otherwise it produces an error.

Input: Model M
Output: Topological sort, $SORT$

```
1  SORT = array [M.attributes()];
2  S = M.getRoots();
3  while |S| > 0 do
4  |   a = S.pop();
5  |   SORT.append(a);
6  |   for (a,b), e = a ↦ b do                    /* For each edge from a to b */
7  |   |   a.removeChild(b);
8  |   |   b.removeParent(a);
9  |   |   M.removeEdge(a,b);
10 |   |   if |b.getParents()| == 0 then          /* b has no more parents */
11 |   |   |   S.add(b);
12 |   end
13 end
14 if |M.getEdges()| > 0 then                     /* M has non covered edges */
15 |   return error;
16 else
17 |   return SORT;
18 end
```

The algorithm for topological sorting is needed due to the fact that the extension of full hierarchies (described in Section 4.1) is included in the implementation. The extension allows for the models to have attributes that influence more parent attributes — have more than one parent. If this was not the case and such situations would not be permissible, then such checking would not be needed. In this case, the model would always represent a tree.

6.5 Scales

Each attribute in DEXx method has a scale assigned to it. The scale is implemented via a class *QScale*. The implementation allows for three different scale types: integer, real or qualitative. In the case of numeric scales DEXx implementation allows for definition of the higher and the lower limit of the scale, where the *high* limit must be at least as high as *low*. Here *low* and *high* are both of type *QIntegerValue* or *QDoubleValue*, in case of integer scale or real scale, respectively. In the case of qualitative scales, the user needs to specify the whole scale. These scales are lists of values, of type *QQualitativeValue*. There is a restriction that no two values must be the same.

All three types of scales allow for definition if the scale is preferentially ordered in an increasing or decreasing order, or if it is unordered. Typically, numeric scales are increasingly ordered (more is better) for performance attributes and decreasingly ordered (less is better) for price-like attributes.

QScale implements the *Shared* interface which allows for usage of the same scale by more than one attribute (QVariable) — for that purpose, QVariable implements the *User* interface. When a scale will be used by more than one attribute, it is set as shared. After that, other attributes may be assigned the same scale, and by that, the attributes are denoted as users of the scale. The scale is now a single object which interacts with multiple attributes and possible corresponding aggregation functions. During the modeling phase, any corresponding *user* attribute may change the properties of the scale (such as type and ordering) — however each change (*Operation*) is firstly checked with all other user attributes, if such action is allowed.

6.5.1 Extended Domain

Extended domains of values are an implicit construct, which is not defined on the level of the attribute. Rather the extended domains (value distributions) are used in alternative's input values and in outputs of the aggregation functions. This inherently means that scales of attributes cannot contain values from the extended domain. Also it means that aggregation function's inputs must be defined for simple value types, not for values from the extended domain.

There are four different value types in the extended domain whose parent class is *QComplexValueType*. Each type of the extended domain stores a set of values from *QSimpleValueType*, where all instances are of the same type.

Together with the values, the *QFuzzyValue* (used for storing fuzzy sets) also stores a mapping of each value to the membership function in the particular fuzzy set. Each value of the membership function must be in the range $(0, 1]$. Here notice that 0 is not included, because by definition, if the membership is 0, then the value is not present in the fuzzy set. Furthermore, the *QDistributionValue* is used for representing probabilistic distributions of values. This complex value type inherits properties of the QFuzzyValue and additionally requires that all membership values (called probabilities) sum up to 1. The library also checks that during every atomic operation on a QDistributionValue again results in a distribution value, otherwise the value is super-classed to QFuzzyValue. QDistributionValue value also supports definition of normally distributed real values. This distribution is sampled m times, where m is a user argument. After that, the original distribution is considered as a discrete distribution.

Two similar types from the extended domain are the *QIntervalValue* and *QSetValue*. Internally, the QIntervalValue stores the lower and the higher limit of the interval, whereas the QSetValue stores all values in the set. However, there is another assumed property for QIntervalValue: the values between the lower and the higher limit are implicitly included.

In contrast, values in the QSetValue can arbitrarily be taken from some scale D and are all explicitly included in the value set.

For each value V in the extended domain a function is in place that checks if the respective value is indeed taken from a scale D. In the case of qualitative scale D, the check is that all values in the complex value V are in the domain D:

$$\forall v \in V \implies v \in D. \tag{6.1}$$

In case of numeric scale D and complex value V, the check is that for each value $v \in V$, v is between the lower limit of D and the higher limit of D:

$$\forall v \in V \implies v \geq D.low \wedge v \leq D.high. \tag{6.2}$$

The algorithm *simplify* (see Algorithm 4.1) is also implemented, which returns a simplified value from a complex value. It can transform a fuzzy set into a distribution if the membership values sum up to 1. It can transform a distribution into a set, if all the probabilities are equal for all elements. Furthermore, it can transform a set into an interval value, if the values indeed form an interval. Finally, it can create a crisp value, if the set size is 1.

Furthermore, the reverse of the *simplify* mechanism is also in place. In some cases there is a need to transform a simpler value into a more complex value. Most notably, this is needed while evaluating aggregation functions on complex values. At that point all values are transformed to either QDistributionValue of QFuzzyValue (based on the DM's preference). Then for each combination of simple value and the respective probabilities (or memberships), the aggregation function is computed and the multiplied probabilities (or minimized memberships) are added to the result. For details, see Section 4.3 and Equations (4.14) and (4.18).

To transform a crisp value v into an interval, the interval $[v, v]$ is created. For transforming an interval $[l, h]$, the set $\{l, v_{l+1}, \ldots, v_{h-1}, h\}$ is produced, where $v_{l+1}, \ldots, v_{h-1}$ are all values preferentially between l and h. The set $\{v_1, v_2, \ldots, v_n\}$ is transformed into a probability distribution $\{v_1/\frac{1}{n}, v_2/\frac{1}{n}, \ldots, v_n/\frac{1}{n}\}$. The probabilistic distribution $\{v_1/\frac{1}{n}, v_2/\frac{1}{n}, \ldots, v_n/\frac{1}{n}\}$ can then also be interpreted as a fuzzy set and is hence not transformed.

6.6 Aggregation Functions

The implementation of DEXx supports five basic (*primitive*) functions, described in Section 4.2 and additionally a general aggregation function. During the evaluation procedure, the aggregation function f_x has access to whole alternative a. It reads the needed input values from the descendant evaluation (or input values) of the aggregated attribute x.

Function type 1 ($QQ \mapsto Q$) is implemented in the *QQualitativeToQualitativeFunction* class. Its internal representation needs to map n input values (from n attributes $a_1, a_2, \ldots, a_n$) to the aggregated qualitative value v in the domain of attribute a. For that purpose we internally used a *Map* object *mapping*, which maps from a *Long* type into an instance of *QValue*. While defining the aggregation function by points, the DM specifies values $(v_{i_1}, v_{i_2}, \ldots, v_{i_n})$, where i_j is the index of the qualitative value in scale D_{a_j}. He/she also specifies the output value $v \in ED_a$. Now, prime numbers $p_1, p_2, \ldots, p_n$ are determined, and a *key* for *mapping* is determined:

$$key = \prod_{j=1}^{n} p_j^{i_j}. \tag{6.3}$$

The *key* is guaranteed to be unique for a unique combination of input values, because such procedure defines an integer number with a unique factorization. Finally, the *key* is assigned to map to the value v. While evaluating such functions with n values, again Equation (6.3) is used to get the *key* of the combination, and the returned value v is retrieved from the *mapping*.

Similarly to *Function type 1*, *Function type 2* ($QQ \mapsto N$) is modeled in *QQualitativeToQuantitativeFunction*. The only difference in the implementation is that the expected aggregated attribute's scale D_a is numeric.

The implementation of *Function type 3* ($NN \mapsto N$) requires to somehow implement mathematical operations into the library. It is implemented in *QQuantitativeToQuantitativeFunction*. For that, we implemented the first stages of the compiling process of Java. This includes lexical analysis, syntactic analysis, semantic analysis and abstract syntax tree creation. The inner *programming language* recognizes numeric constants, mathematical operations ($+$, $-$, $\cdot$, $/$, $==$, $!=$, $<=$, $>=$, $<$, $>$) and four Java functions: sqrt, power, log and random. Furthermore, we also allow logic constants true and false, and logic operations , && and ||. Additionally, the control statements similar to Java's for, and if and else are also supported. For the function to be complete, it also recognizes symbols \$1, \$2, ..., which represent positional argument values to the function.

The internal programming language accepts a string, for example return $power(\$1, \$2)$ (raising the first argument to the power of the second argument). The first point of the compiler parses the string for lexems, "return", "power", "(", "\$1", ",", "\$2" and ")". It checks that the input string is syntactically correct. After that, it checks if the return statement is present and that power function indeed returns a numeric value and that arguments to power function are numeric values — in semantic analysis. Finally, it constructs an abstract syntax tree, which is used for evaluation of the function. Simply starting at the root of the syntax tree, the tree is evaluated depth-first and values of \$1 and \$2 are replaced by the respective input values for evaluation. If at any point, the analysis of the function does not succeed, or during evaluation an exception is raised, the *AggregationFunctionException* is returned to the user.

The compiling process implemented for function *QQuantitativeToQuantitativeFunction* allows for even more design freedom — the function together with integer and double types, also supports string inputs. This means it can be used as a general function of qualitative and quantitative attributes, and can model even more complex control flow operations than just numeric value manipulation.

To implement the *Function type 4* ($N \mapsto Q$), internally a *Map* object is used, called *mapping* (*QQuantitativeToQualitativeFunction*). The DM specifies pairs of type $([l, h], v)$, where $[l, h]$ is an instance of $QIntervalValue$, and values l and h are respective lower and higher numeric limits of the interval. v is the qualitative value, expected as the output. A *List list* object is also used to store the intervals. During the definition of the aggregation function, the interval $[l, h]$ is added to the *list* and also v is put into the *mapping* as a value with $[l, h]$ as the key. While evaluating, for a given numeric input m, the *list* object is traversed, and checked if $m \in [l_i, h_i]$, until such interval is found. Finally, the value v is returned, which is defined for the $[l_i, h_i]$ in the *mapping*. This procedure allows the DM to specify non-disjoint intervals at the input. The function outputs the first value v for which the value m is in the respective interval. If there are no matching intervals for the input value m, the *AggregationFunctionException* is raised to the user.

To model the aggregation *Function type 5* ($NQ \mapsto N$) ideas are borrowed from implementation from *Function type 1* and *Function type 4* — it is implemented in *QQuantitativeQualitativeToQuantitativeFunction*. Internally it uses the same qualitative *mapping* object, except it maps to objects of *QQuantitativeToQuantitativeFunction*, called *numericFunc-*

tion. The *mapping* is defined on all qualitative input attributes and *numericFunction*s are defined on all numeric attributes to the aggregated attribute a. Definition of the function is by pairs $((v_{i_1}, v_{i_2}, \ldots, v_{i_n}), numericFunction)$. The qualitative values $(v_{i_1}, v_{i_2}, \ldots, v_{i_n})$ are used to determine the *key* for the *mapping*, using Equation (6.3), and the value assigned to the key is *numericFunction*. During evaluation, the qualitative input values $(v_{i_1}, v_{i_2}, \ldots, v_{i_n})$ determine the *key* for the *mapping*, and the respective *numericFunction* is used to produce the final result on the remaining numeric input values.

The abstract implementation of general aggregation functions (*QGeneralAggregationFunction*) supports implementation of application specific functions, which may be needed to compute various functions. Such examples are querying a database or making a call to an external web-service. The function is implemented by the programmer of the application (or the DM). The function must be sub-classed from the QGeneralAggregationFunction and must implement two methods: *isComputable* and *computeValue*. Method *isComputable* queries the function if in its current state, the function is computable for any combination of input values. Method *computeValue* is the core body of the function, and must hence produce a value (from the scale of the aggregated attribute) for given input values (taken from the scales of the input attributes).

Further, if such a class is shared and *toXML* and *fromXML* methods are also implemented, then such a function can be shared to other users of the methodology. Even more, the exported models using such a function can also be shared.

6.6.1 Relational Aggregation Functions

Relational aggregation functions, implemented in *QRelationalAggregationFunction* internally hold a reference to a predefined set of allowed operations. The aggregation function computes an operation on evaluations of relational alternatives. The operations on arbitrary number of arguments are:

- *distribution* computes the distribution over the values appearing in the evaluations.
- *interval* creates an interval from the lowest evaluation to the highest possible evaluation, among all evaluations.
- *least_frequent* computes the least frequent element appearing in the evaluations.
- *least_frequent_all* computes the least frequent element appearing in the evaluations and including in the scale of the output attribute.
- *max* computes the preferentially maximal value appearing in the evaluations.
- *mean* computes the numeric mean value of the appearing evaluations.
- *median* computes the numeric median value of the appearing evaluations.
- *min* computes the preferentially minimal value appearing in the evaluations.
- *mode* computes the numeric mode value of the appearing evaluations.
- *product* computes a numeric product of the evaluations.
- *random* returns a random value appearing in the evaluations.
- *set* computes a set of all values appearing as evaluations of relational alternatives.
- *sum* computes a numeric sum of the input evaluations.

- *weighted_ avg* computes the numeric weighted average of numeric evaluations. Here a second relational input is expected, which gives the value of the weight.
- *weighted_ distribution* computes a distribution of values, considering an additional relational input as the weight.

Furthermore, the *QRelationalAggregationFunction* can be sub-classed, so that a custom aggregation function can be defined. Such function is presented later in Algorithm 7.2.

To evaluate a relational aggregation function, firstly all relational alternative evaluations are computed for the counterpart relational inputs, and then the selected operation is applied on the values of the evaluations.

6.7 Alternatives

The alternatives in DEXx are implemented using class *QAlternative*. Internally, the alternatives hold a reference to all attributes (QVariable), differentiating between regular input attributes, relational attributes and other aggregated attributes. While evaluating, the evaluations of aggregation functions are written into the respective variables of the alternative. Internally, the alternatives store evaluations in a *Map* object, which is mapping from the QVariable instance, to an instance of QValue, v. Before actually writing the evaluation for some attribute a, the alternative checks if the evaluation is indeed taken from the scale of attribute a, $v \in ED_a$.

Similarly, *QRelationalAlternative* stores all attribute references for the respective relational model, the alternative belongs to. Additionally, the relational alternatives store a reference to their parent (relational) alternative.

6.8 Models

QModel is the main class which guides the model development and alternative evaluation in DEXx. The model holds a reference to all used variables, specifically distinguishing the input and output variables, and alternatives. The model is the *Owner* of all present attributes.

The model checks the integrity of the model by checking with the appropriate methods implemented in the classes. For example, it repeatedly checks for cycle existence (using Algorithm 6.1) after each addition of a connection between two attributes (update to successor relation S).

Furthermore, before the evaluation of alternatives, the model checks if all alternatives have an input assigned to every input attribute, and that all involving aggregation functions are computable (using the method *isComputable*). The main model also makes a check for all relational attributes, if a relational aggregation function is defined and that all relational models involved also have at least one relational alternative defined.

QModel mainly communicates with the attributes, scales, aggregation functions and relational models via so-called *Operation*s. Every time an attribute wants to change (for example its name), change its scale or aggregation function, it requests so with a *canChange* operation to the model. The model may respond with an operation *canChange* or *noChange*. The model issues the signal based on its internal state — for example, it may be locked for changes and hence issue a *noChange* signal. If the signal from the model is *canChange*, then the attribute can make the proposed changes and after that, it again issues a signal *changed* to the model. This signal notifies the model about the change, so that the model can update any internal data structures which it uses. In the case, the model does not allow a change, a *GeneralModelException* is raised.

Similarly to QModel, the *QRelationalModel* also communicates with its attributes, scales, aggregation functions and other relational models. Additionally, it also communicates with its parent model for changes the relational model may want to make.

6.9 Evaluation of Alternatives

The model guides the evaluation of alternatives. The implemented evaluation of alternatives is influenced by the Algorithm 4.2. The evaluation of an alternative a is done in a concurrent fashion — aggregation functions that are independent are run concurrently. The model spawns a *thread* for each aggregation function and puts the thread to sleep. When all inputs for the particular aggregation function are available, it runs the thread, and terminates it, when the aggregation function evaluation is produced. When a relational aggregation function f_{ra} is encountered during evaluation, the evaluation procedure is called recursively on the respective relational model RM with the relational alternatives $RA = a.relationalAlternatives(RM)$, corresponding to the main alternative a. The evaluation of f_{ra} is then put to sleep, until all evaluations $V = (V_1, V_2, \ldots, V_{|RA|})$ of relational alternatives RA are available. Finally, thread for f_{ra} is run with evaluations V.

The implementation allows for full evaluation of alternatives and even a partial evaluation — this is particularly useful, if the user is not interested in the evaluation of the whole model, rather he/she is interested in an evaluation of a specific alternative on a given attribute.

6.10 Important Methods on Models

To simplify the model development process and to provide additional means of model representation we developed five important methods. These methods are:

createWebPage method supports creation of static web-pages, formatted using *Bootstrap* framework — supporting styling of the web-pages. The method exports numerous HTML files which are already formatted to the extent, where they can be published online. The web-pages include general information about the model (name, description, number of attributes, aggregated attributes and root attributes). It also has space for further manual development of the web-page, for example, when one needs to describe his/her project in more detail. Furthermore, the web-pages export the structure of the model as a dynamic table, so that the user can navigate through the model with ease. Each attribute in the model-view includes its description, scale, (possible) aggregation function and its direct descendants. The aggregation functions are exported as tables that can be dynamically opened. All present alternatives are exported on their own page, displaying their respective input values. Another page for the evaluation of present alternatives is exported, which enables the DM to navigate the evaluations of alternatives at every level of the model. The last exported page constructs a HTML form, which lists all input attributes and alongside also their scales as drop-down menus. This is particularly useful, if the DM wants to enable the users of the web-page to evaluate their own alternatives with the library, using an exposed web-service.

The method was actively used for viewing and discussing large amounts of produced evaluations during the sustainability assessment of electric energy production technologies use-case, presented in Chapter 9.

exportToGRAPHML method allows for exporting the model structure to a standard *GRAPHML* format. The method exports all attribute names and their respective

descriptions. It generates all links (including hierarchical) using GRAPHML syntax. The exporting procedure additionally colors the root, aggregated, input and relational attributes with different colors. Furthermore, it also exports all relational models recursively into the same file, while encapsulating the relational model as a separate component. The produced file can be further viewed an analysed by applications, such as *yED Graph Editor*.

This method was used for viewing the structure of the large models produced during the sustainability assessment of electric energy production technologies use-case, presented in Chapter 9.

exportXML method exports models to a XML-formatted file. This is particularly usable for reusability of developed models in different applications. Together with the model and relational model structure, all attribute names, descriptions and scales are exported. Additionally, the aggregation functions for aggregated attributes are exported in human readable format. This allows for later changing of the aggregation functions in the XML files. Finally, if there are any alternatives defined for the model, they are exported with their respective input values, their name and description. Additionally, if alternatives have relational alternatives assigned, they are similarly exported and hold a reference to the parent alternative.

Method is usually used for saving finished models, or models in development. Particularly, the method was used in all four use-cases presented in Chapters 7, 8, 9 and 10.

importFromDEXi method allows for reading models and alternatives developed using DEXi (Bohanec, 2015b, 2014). The method reads all information about model attributes — names and description, connections between attributes and also induces the *links* between hierarchically connected attributes. Furthermore, the procedure induces the aggregation functions from the DEXi encoded aggregation function descriptions. Finally, the method also creates all alternatives present in the file. All read objects are translated to the DEXx context and allow for immediate continuation of the development/evaluation process. Also note, the models imported with *importFromDEXi* can be saved in more powerful XML format with *exportXML*.

This method is used during translation of the DEXi format models to DEXx models. The method was used in all four use-cases presented in Chapters 7, 8, 9 and 10.

importXML is a method which reads the XML representations of models exported with *exportXML*, or developed by hand in XML with the correct format. Similarly to *importFromDEXi* the objects present in the XML format are translated to objects in DEXx, and can be used for further development/evaluation.

Method is used for restoring the saved models. Particularly, the method was used in all four use-cases presented in Chapters 7, 8, 9 and 10.

The implemented library support some dynamic aspects of model development. These dynamic aspects include:

- Changing the order of values in attribute scales. The order does not influence the computation of respective aggregation functions, however it may yield aggregation function being non-consistent. The computation of the *key* for the purposes of evaluation is independent of the value position in the scale.

- Changing the order of input attributes to some aggregated attribute. The order of attributes does not influence the evaluation of aggregation functions, because each attribute has a prime number assigned which is positionally independent (see Equation (6.3)).
- Changing the order of the scale from increasing/decreasing to decreasing/increasing. Similarly to the first point, the *key* for a particular value is independent of the value's position in the scale.
- Dynamic evaluation of model alternatives allows for evaluation only of parts of the model for a given alternative, if the alternative has already been evaluated. In essence, the models in DEXx additionally hold information if the sub-hierarchy of the model has been changed. If a part of the model has not changed from the previous evaluation of the alternative, then it is not necessary to evaluate the alternative on the particular part — rather the value can be read directly from the alternative. On the other hand, if at some point the model has changed, the alternative must again be evaluated on this particular part.

The library however does not currently support the *dynamic* aspects of modeling. We must stress that by "support" we are referring to preserving and manipulating information the DM entered in the model. The library does not support dynamic function changing while a value for some attribute is *added* or *deleted*. Also, the existing aggregation functions cannot cope with their representation change, while attributes are being added or deleted from the respective aggregated attribute.

One of the lacking features in the implemented library is the possibility to analyse models for their consistency and sensitivity. Another useful feature which is lacking in the library is the pairwise (or multiple-wise) comparison of alternatives — enabling the DM to compare the evaluated alternatives on lower-laying levels or by their respective strong and weak components.

Chapter 7

Model for Assessment of Public e-Portals

This chapter is the first of the four chapters presenting four use-cases which were implemented using the DEXx methodology and the developed implementation. The use-cases show that development of complex models, which were previously hard or even impossible to implement, is now natively supported. Decision problems addressed in these scenarios have properties that, in order to be successfully solved, require the methodological extensions developed in this thesis. The scenarios involve qualitative and numerical attributes which form a full hierarchy. The values of the attributes are not just crisp values, but also include distributions, intervals and sets. On top of that, relational aggregation of relational alternatives are required. All aggregation is performed with numeric and/or qualitative values using some predefined function type or a custom aggregation function.

The four use-cases cover different real-life decision problems:

- Assessment of Public e-Portals: A qualitative relational model with a custom aggregation function for better representation of complex parts of a model on the domain of assessment of public e-portals. The use-case is presented in this chapter.
- Assessment of Bank Reputational Risk: A qualitative and quantitative relational model with high data input processing capabilities on the domain of assessing bank reputational risk. See Chapter 8.
- Sustainability Assessment of Electric Energy Production Technologies: A qualitative and quantitative relational model with additional capability for simulating the model through time on the domain of assessing sustainability of energy options in Slovenia. See Chapter 9.
- Water flows in Agriculture: A qualitative model with regular alternatives on the domain of evaluation of water flows in agriculture. See Chapter 10.

Electronic administration portals are a platform for electronic administrative services for different users (citizens and businesses) (Leben & Bohanec, 2004; Leben et al., 2006). The quality and user-friendliness of these services depends to a large extent on how well planned and developed it is. The study for evaluating e-portals was carried out at the Faculty of Administration, University of Ljubljana, Slovenia.

The study for evaluation of e-portals required an explicit treatment of relational alternatives and relational models. This was achieved by developing models using DEXi (Bohanec, 2015b), evaluating the models, and large amounts of manual intervention for computation of relational input values.

The purpose of this use-case presentation is to repeat the study, by implementing the models using DEXx and the developed implementation, in a native way. The newly implemented case study uses the native support for relational models and relational alternatives, with small usage of numeric attributes. The information about the models and alternatives has been retrieved from authors Leben and Bohanec (2004).

The results produced using DEXx show that study of e-portals can be indeed implemented with the DEXx library, producing identical results as the original study, and without requiring any manual work during evaluation of alternatives. This fact justifies the inclusion of the relational models and relational alternatives into the DEXx method.

In this use-case we reconstructed the model with DEXx and reimplemented the model using the developed implementation. The relational aggregation functions were implemented using DEXx, which in turn eliminated the need for manual relational aggregation. With this setting, all the evaluations were computed automatically, without the need of manual intervention and other software. All the produced results were the same as in the original study (Leben & Bohanec, 2004; Leben et al., 2006).

7.1 Problem Description

The models were required to measure the level of sophistication, coverage, coordination and accessibility of a service, and combine these into an overall portal score, considering the life-events that the users of the models could be in. From these requirements, authors deduced that three different models were needed. Additionally, the models needed to be connected between them, where evaluations of lower level alternatives needed to be aggregated — and their aggregate evaluations used in the higher level models. The three models were:

1. Model for *Portal* evaluation. This model evaluated the portal over-all considered attributes and supported life events. Additionally, the model covered the usability of the portal and coverage of the life events.

2. Model for *Life event* evaluation. Model evaluates a specific life event, considering all services supporting this particular life event. Even more, the model considers the maturity of the supported life event, its usability and clarity of the life event.

3. Model for *E-service* evaluation. This model is at the lowest level of granularity and it evaluates single e-services. It considers the clarity, given information and sophistication of e-service. This model has two root attributes:

   - E-service evaluation is the main evaluation of the e-service, and
   - E-service type is the additional information needed at the Life event evaluation, because the two types of e-services are aggregated differently.

The model for portal evaluation is presented in Table 7.1. The model is used for evaluation of each portal, given the evaluations of supported life events. The root attribute is *Portal*, where the final evaluation of portal is produced. The three main branches of Portal are *LE Solution Methods*, *LE Coverage* and *LE Usability*. Here, LE Solution Methods (written in blue text) is an input attribute. To evaluate this attribute's input aggregation of all Life Events needs to be accomplished (evaluations of all life events assigned to this particular portal). For the relational aggregation function specification, please refer to Section 7.3.1.

The model for life event evaluation is shown in Table 7.2. The model serves as the evaluation tool for life events. Its evaluations are based on the evaluations of e-services

Table 7.1: The model for assessing portals with relational models. On the left side, the structure of the attributes is presented, and on the right side, their corresponding scales are displayed. The scales are ordered from worst values (displayed in red) to best values (displayed in green).

Attribute	Scale
Portal	unacceptable, acceptable, good, very good, excellent
LE Solution Methods	unacceptable, acceptable, good, very good, excellent
Life Event	unacceptable, acceptable, good, very good, excellent
LE Coverage	inadequate, partly-adequate, adequate
Coverage with LEs	inadequate, partly-adequate, adequate
Topics Coverage	inadequate, partly-adequate, adequate
LEP Usability	unsuitable, partly-suitable, suitable, very suitable
Access to LE	unsuitable, partly-suitable, suitable, very suitable
List of LE	inadequate, partly-adequate, adequate
Hierarchy of Topics	inadequate, partly-adequate, adequate
Search Engine	inadequate, partly-adequate, adequate
Standardisation of LE	inadequate, partly-adequate, adequate

supporting this particular life event. Furthermore, the evaluations of life events are used at the higher level model for assessment of the corresponding portal. In this model, the root attribute is *Life Event*. The three main considered sub-trees are *Maturity*, *Usability of LE* and *LE clarity*. Written in blue text (*LE Sophistication*) is the relational attribute that aggregates all e-services supporting this particular life event. The inputs to LE Sophistication (*E-service*, *E-service Type*) are both produced by the model for e-service evaluation.

Table 7.2: The model for assessing life events with relational models. On the left side, the structure of the attributes is presented, and on the right side, their corresponding scales are displayed. The scales are ordered from worst values (displayed in red) to best values (displayed in green).

Attribute	Scale
Life Event	unacceptable, acceptable, good, very good, excellent
Maturity	unacceptable, acceptable, good, very good, excellent
LE Sophistication	unacceptable, acceptable, good, very good, excellent
E-service	unacceptable, acceptable, good, very good, excellent
E-service Type	vital, supplementary
Scope of LE	inadequate, partly-adequate, adequate
Vital Scope	inadequate, partly-adequate, adequate
Supplementary Scope	inadequate, partly-adequate, adequate
LE Coordination	dispersed, one-entry point, step-by-step, one-step
Usability of LE	unsuitable, partly-suitable, suitable, very suitable
Access to Services	unsuitable, partly-suitable, suitable, very suitable
Access to Instruments	unsuitable, partly-suitable, suitable, very suitable
Key Steps	inadequate, partly-adequate, adequate
Check List	inadequate, partly-adequate, adequate
FAQ	inadequate, partly-adequate, adequate
E-guide	inadequate, partly-adequate, adequate
Standardisation of Services	inadequate, partly-adequate, adequate
LE clarity	inadequate, partly-adequate, adequate

The model for e-service evaluation is presented in Table 7.3. The model is used for evaluation of e-services, independent of the corresponding portal and life event. In this model, two root attributes are present *E-service* — giving the evaluation of the e-service, and *E-service Type* — producing the type of this particular e-service (either *vital* or *supplementary*). The e-service evaluation is based on three main components: *Clarity of e-Service*, *Information* and the *Sophistication*. The evaluations of e-services are used in the model for life event evaluation.

Table 7.3: The model for assessing particular e-service with relational models. On the left side, the structure of the attributes is presented, and on the right side, their corresponding scales are displayed. The scales are ordered from worst values (displayed in red) to best values (displayed in green).

Attribute	Scale
E-service	unacceptable, acceptable, good, very good, excellent
Clarity of e-Service	inadequate, partly-adequate, adequate
Information	inadequate, partly-adequate, adequate
Information Quality	inadequate, partly-adequate, adequate
Information Accessibility	inadequate, partly-adequate, adequate
Sophistication	unsuitable, partly-suitable, suitable, very suitable
Documents	unsuitable, partly-suitable, suitable, very suitable
Document Accessibility	unsuitable, partly-suitable, suitable, very suitable
Downloadable Documents	inadequate, partly-adequate, adequate
Download	inadequate, partly-adequate, adequate
Send	inadequate, partly-adequate, adequate
Interaction	inadequate, partly-adequate, adequate
Interactive Forms	inadequate, partly-adequate, adequate
Attachments	inadequate, partly-adequate, adequate
Authentication	inadequate, partly-adequate, adequate
Additional Features	unsuitable, partly-suitable, suitable, very suitable
Notifying	inadequate, partly-adequate, adequate
E-payment	inadequate, partly-adequate, adequate
E-delivering	inadequate, partly-adequate, adequate
E-service Type	vital, supplementary

The specific descriptions of the used attributes and their interpretations are available in (Leben et al., 2006).

Models for portal, life event and e-service assessment are relationally structured as presented in Figure 7.1. The figure shows that multiple evaluations of e-services influence a single life event (through relational attribute rx_L and counterpart relational inputs ox_E). Further, multiple evaluations of life events influence a single portal, through relational attribute rx_P and counterpart input attribute ox_L.

Assessment of portals considers evaluations of corresponding life events. The study employs a translation table, presented in Table 7.4. All life-event evaluations are mapped to an integer value, according to the position in the scale (see Equation (7.1)), for the purpose of relational aggregation. Their average is then mapped to an integer value according to Table 7.4. Finally, the inverse function of *translate* is used to compute the result of the relational aggregation.

$$translate(v) = \begin{cases} 1 & \text{if } v = \text{unacceptable} \\ 2 & \text{if } v = \text{acceptable} \\ 3 & \text{if } v = \text{good} \\ 4 & \text{if } v = \text{very good} \\ 5 & \text{if } v = \text{excellent} \end{cases} \tag{7.1}$$

The assessment of life-events similarly considers evaluations of the corresponding e-services. It employs a similar table for relational aggregation, presented in Table 7.5. The values of e-services are mapped to integer values using function *translate* (see Equation (7.1)). The averages of numeric evaluations, for *vital service score* VS (services that are vital) and for *supplementary service score* SS (services that are supplementary), are computed. The aggregate average is computed using Equation (7.2), mapped to an integer using Table 7.5 and finally to a qualitative value using the inverse of function *translate*.

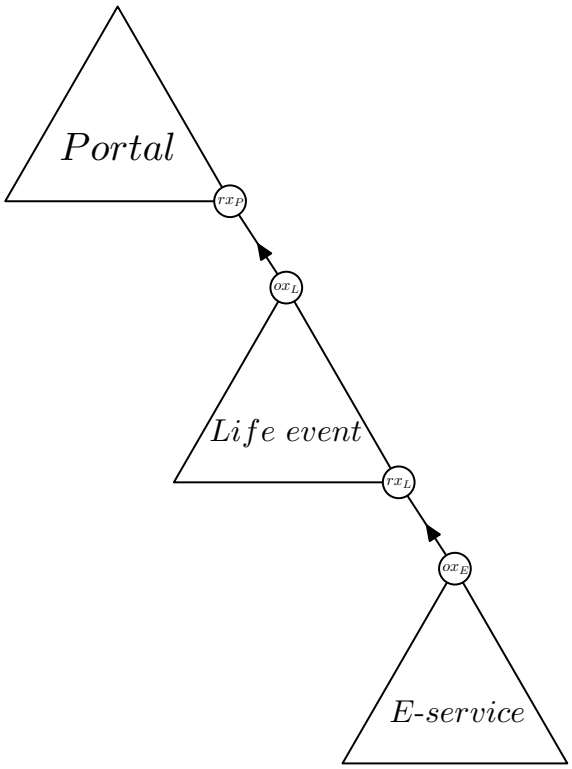

Figure 7.1: The configuration of main model for portal assessment with the relationally connected model for life event assessment, which in turn depends on respective e-service assessment. Multiple e-services influence evaluation of a single life event, and multiple evaluations of life-events influence a single portal.

Table 7.4: The translation table from average life-event score to the relational value of LE solution methods.

Average LE score	LE solution methods
from 1 to 1.4	1
from 1.5 to 2.3	2
from 2.4 to 3.3	3
from 3.4 to 4.2	4
from 4.3 to 5	5

$$aggregate(VS, SS) = \begin{cases} VS & \text{if } SS = 0, \\ \frac{2}{3} + \frac{1}{3}SS & \text{if } VS = 0, \\ \frac{2}{3}VS + \frac{1}{3}SS & \text{otherwise.} \end{cases} \tag{7.2}$$

In the original study, the three models were developed using DEX method (Bohanec & Rajkovič, 1990, 1999; Bohanec, Rajkovič, et al., 2013) and DEXi software (Bohanec, 2015b). The evaluation in the study was done semi-automatically. All e-services were evaluated using DEXi, then all evaluations were manually mapped to the corresponding life events. The relational aggregation of e-services was done using Microsoft Excel. The produced relational evaluations were introduced to DEXi for assessment of life-events. Life events were again evaluated using DEXi, mapped to the corresponding portals and their

Table 7.5: The translation table from weighted e-service score to the relational value of LE sophistication.

Average e-service score	LE sophistication
from 1 to 1.7	1
from 1.8 to 2.5	2
from 2.6 to 3.3	3
from 3.4 to 4.1	4
from 4.2 to 5	5

evaluations relationally aggregated using Microsoft Excel. The evaluations of portals were finally produced by inputting the life-event relational evaluations into DEXi.

7.2 Model Description and Structure

The model structure for Public e-Portal evaluation was reconstructed from the study (Leben & Bohanec, 2004; Leben et al., 2006). The models for portal, life event and e-service evaluation are presented in Tables 7.1, 7.2 and 7.3, respectively.

The models implemented using DEXx have smaller differences added, in order to model the relational aggregation functions precisely as in the original study. The main problem here was the inclusion of numeric attributes, since the evaluation procedure in the original study considered numeric values in an implicit way (see Tables 7.4 and 7.5). With DEXx, the numeric attributes need to be explicitly defined. The key differences are:

Portal evaluation has one attribute added as a child to the *LE Solution Methods* attribute. We call this attribute *AVG Life Event* and its scale are positive real numbers. AVG Life Event is the new relational attribute in model for portal evaluation, which represents the average evaluation value over all life events supported by a particular portal. Its counterpart relational input is the new attribute called *N Life Event*, which is described below.

Life Event evaluation model has two changes:

- A new attribute called *N Life Event* is added to the model. The attribute is placed as a root in the model. Its input is the previous root of the model (*Life Event*). The scale of the attribute are all integers from 1 to 5 inclusively. The attribute represents the numerical evaluation of a particular life event.
- A new relational attribute called *AVG E-service* is added to the model. The attribute is placed as an input to *LE Sophistication*. The relational inputs to the attribute are *N E-service* and *E-service Type*. The scale of AVG E-service are positive real numbers. The attribute will compute the aggregated life event sophistication based on Equation (7.2).

E-service evaluation has one attribute added as a root of the model. The attribute is called *N E-service* and represents the numeric value of the final evaluation of the e-service. Its input is attribute *E-service*. The scale of the new attribute are all integers from 1 to 5 inclusive.

Formally, model for evaluation of portals has the following attributes (the equations are derived from Table 7.1):

$$X = \{\text{Portal}, \text{LE Solution Methods}, \text{LE Coverage}, \ldots, \text{Standardisation of LE}\}. \tag{7.3}$$

The descendants of attributes are described by mapping S (see Equation (3.4)):

$$\begin{gathered} S(\text{Portal}) = \{\text{LE Solution Methods}, \text{LE Coverage}, \text{LEP Usability}\}, \\ S(\text{LE Coverage}) = \{\text{Coverage with LEs}, \text{Topics Coverage}\}, \\ S(\text{LEP Usability}) = \{\text{Access to LE}, \text{Standardisation of LE}\}, \\ S(\text{Access to LE}) = \{\text{List of LE}, \text{Hierarchy of Topics}, \text{Search Engine}\}, \\ S(\text{LE Solution Methods}) = \{\text{AVG Life Event}\}, \\ S(\text{Coverage with LEs}) = \{\}, \\ \ldots, \\ S(\text{Standardisation of LE}) = \{\}. \end{gathered} \tag{7.4}$$

Now, also the $aggAttributes_{\text{Portal}}$ can be defined as:

$$\begin{gathered} aggAttributes_{\text{Portal}} = \{\text{Portal}, \text{LE Coverage}, \text{LEP Usability}, \text{Access to LE}, \\ \text{LE Solution Methods}\}. \end{gathered} \tag{7.5}$$

Furthermore, $modelInputs_{\text{Portal}}$ can be computed:

$$\begin{gathered} modelInputs_{\text{Portal}} = \{\text{AVG Life Event}, \text{Coverage with LEs}, \text{Topics Coverage}, \\ \text{List of LE}, \text{Hierarchy of Topics}, \text{Search Engine}, \text{Standardisation of LE}\}. \end{gathered} \tag{7.6}$$

Finally, the computed $modelOutputs_{\text{Portal}}$ are:

$$modelOutputs_{\text{Portal}} = \{\text{Portal}\}. \tag{7.7}$$

The scales of the attributes are also shown on the right side of the Table 7.1. The scales to the corresponding attributes are:

$$\begin{gathered} D_{\text{Portal}} = (\text{unacceptable}, \text{acceptable}, \text{good}, \text{very good}, \text{excellent}), \\ D_{\text{LE Solution Methods}} = (\text{unacceptable}, \text{acceptable}, \text{good}, \text{very good}, \text{excellent}), \\ \ldots, \\ D_{\text{Standardisation of LE}} = (\text{inadequate}, \text{partly-adequate}, \text{adequate}), \\ D_{\text{AVG Life Event}} = \mathbb{R}^{+}. \end{gathered} \tag{7.8}$$

Moreover, since the model for evaluation of portals has a subordinate relational model, it also has a relational attribute (rx): *AVG Life Event*. Its counterpart attribute (ox) from the relational model for life event evaluation is *N Life Event*.

Similarly, attributes X, successors S, aggregated attributes $aggAttributes_{\text{Life Event}}$, model inputs $modelInputs_{\text{Life Event}}$, model outputs $modelOutputs_{\text{Life Event}}$ and scales D, are constructed from Table 7.2 for the model for life event assessment.

$$X = \{\text{Life Event}, \text{Maturity}, \text{LE Sophistication}, \ldots, \text{LE clarity}\}, \tag{7.9}$$

$$\begin{gathered} S(\text{Life Event}) = \{\text{Maturity}, \text{Usability of LE}, \text{LE clarity}\}, \\ S(\text{Maturity}) = \{\text{LE Sophistication}, \text{Scope of LE}, \text{LE Coordination}\}, \\ \ldots, \\ S(\text{Access to Instruments}) = \{\text{Key Steps}, \text{Check List}, \text{FAQ}\}, \\ S(\text{LE Sophistication}) = \{\text{AVG E-service}\}, \\ S(\text{AVG E-service}) = \{\}, \\ S(\text{Vital Scope}) = \{\}, \\ \ldots, \\ S(\text{LE clarity}) = \{\}, \end{gathered} \tag{7.10}$$

$$\begin{gathered} aggAttributes_{\text{Life Event}} = \{\text{Life Event}, \text{Maturity}, \text{Scope of LE}, \text{Usability of LE}, \\ \text{Access to Services}, \text{Access to Instruments}, \text{LE Sophistication}\}, \end{gathered} \tag{7.11}$$

$$\begin{gathered} modelInputs_{\text{Life Event}} = \{\text{AVG E-service}, \text{Vital Scope}, \text{Supplementary Scope}, \\ \text{LE Coordination}, \text{Key Steps}, \text{Check List}, \text{FAQ}, \text{E-guide}, \\ \text{Standardisation of Services}, \text{LE clarity}\}, \end{gathered} \tag{7.12}$$

$$modelOutputs_{\text{Life Event}} = \{\text{Life Event}\}, \tag{7.13}$$

$$\begin{gathered} D_{\text{Life Event}} = (\text{unacceptable}, \text{acceptable}, \text{good}, \text{very good}, \text{excellent}), \\ D_{\text{Maturity}} = (\text{unacceptable}, \text{acceptable}, \text{good}, \text{very good}, \text{excellent}), \\ \ldots, \\ D_{\text{LE clarity}} = (\text{inadequate}, \text{partly-adequate}, \text{adequate}), \\ D_{\text{AVG E-service}} = \mathbb{R}^+, \\ D_{\text{N Life Event}} = [1, 5]. \end{gathered} \tag{7.14}$$

Again, the model for evaluation of life events has a subordinate relational model, it also has a relational attribute (rx): *AVG E-service*. Its counterpart attributes (ox) from the relational model for e-service evaluation are *N E-service* and *E-service Type*.

Attributes X, successors S, aggregated attributes $aggAttributes_{\text{E-service}}$, model inputs $modelInputs_{\text{E-service}}$, model outputs $modelOutputs_{\text{E-service}}$ and scales D, are constructed from Table 7.3 for the model for e-service assessment.

$$\begin{gathered} X = \{\text{N E-service}, \text{E-service}, \text{Clarity of e-Service}, \\ \text{Information}, \ldots, \text{E-service Type}\}, \end{gathered} \tag{7.15}$$

$$\begin{gathered} S(\text{N E-service}) = \{\text{E-service}\}, \\ S(\text{E-service}) = \{\text{Clarity of e-Service}, \text{Information}, \text{Sophistication}\}, \\ S(\text{Information}) = \{\text{Information Quality}, \text{Information Accessability}\}, \\ \ldots, \\ S(\text{Additional Features}) = \{\text{Notifying}, \text{E-payment}, \text{E-delivering}\}, \\ S(\text{Information Quality}) = \{\}, \\ S(\text{information Accessability}) = \{\}, \\ \ldots, \\ S(\text{E-service Type}) = \{\}, \end{gathered} \tag{7.16}$$

$$aggAttributes_{\text{E-service}} = \{\text{N E-service, E-service, Information, Sophistication, Documents, Document Accessability, Downloadable Documents, Interaction, Additional Features}\}, \tag{7.17}$$

$$modelInputs_{\text{Life Event}} = \{\text{Information Quality, Information Accessability, Download, Send, Interactive Forms, Attachments, Authentication, Notifying, E-payment, E-delivering, E-service Type}\}, \tag{7.18}$$

$$modelOutputs_{\text{E-service}} = \{\text{N E-service, E-service Type}\}, \tag{7.19}$$

$$\begin{gathered} D_{\text{N E-service}} = [1, 5], \\ D_{\text{E-service}} = (\text{unacceptable, acceptable, good, very good, excellent}), \\ D_{\text{Clarity of e-Service}} = (\text{inadequate, partly-adequate, adequate}), \\ \ldots, \\ D_{\text{E-service Type}} = \{\text{vital, supplementary}\}. \end{gathered} \tag{7.20}$$

Because the model for e-service evaluation does not have any subordinate relational models, there are no relational attributes and no relationally connected counterpart attributes.

7.3 Aggregation Functions

We present the aggregation functions for selected attributes for each of the models, which are derived from the DEX models used in the original study (Leben & Bohanec, 2004; Leben et al., 2006).

The aggregation function for Portal attribute in the portal evaluation is given in Table 7.6. The aggregation function aggregates three input attribute values of *LE Solution Methods*, *LE Coverage* and *LEP Usability* into the evaluation of *Portal*. First three columns represent all possible combinations of input values, and the right-most column gives the output value for the corresponding combination. Each combination row in the table is read as a rule. For example, the first line reads: `if` LE Solution Methods is *unacceptable* `and` LE Coverage is *inadequate* `and` LEP Usability is *unsuitable* `then` Portal is *unacceptable*. Similarly, all other rules can be interpreted.

Aggregation function for *LE Solution Methods* is given in Equation (7.21). It aggregates a numeric value ($D_{\text{AVG Life Event}}$) to a qualitative value (from $D_{\text{LE Solution Methods}}$), according to Table 7.4. The function is of primitive type 4 (see Equation (4.5)).

$$NQ_{\text{LE Solution Methods}}(y) = \begin{cases} \text{unacceptable} & \text{if } y \in [1, 1.4] \\ \text{acceptable} & \text{if } y \in (1.4, 2.3] \\ \text{good} & \text{if } y \in (2.3, 3.3] \\ \text{very good} & \text{if } y \in (3.3, 4.2] \\ \text{excellent} & \text{if } y \in (4.2, 5] \end{cases} \tag{7.21}$$

N Life Event aggregation function transforms a qualitative value into a numeric integer value, and is hence a *Function type 2*. Its purpose is to prepare the evaluations of life events for their averaging in the relational attribute *AVG Life Event*. It aggregates qualitative values from $D_{\text{Life Event}}$ into domain $D_{\text{N Life Event}}$. It is defined as:

Table 7.6: The aggregation function for Portal evaluation from LE Solution Methods, LE Coverage and LEP Usability.

LE Solution Methods	*LE Coverage*	*LEP Usability*	*Portal*
unacceptable	inadequate	unsuitable	unacceptable
unacceptable	inadequate	partly-suitable	unacceptable
unacceptable	inadequate	suitable	unacceptable
unacceptable	inadequate	very suitable	unacceptable
unacceptable	partly-adequate	unsuitable	unacceptable
unacceptable	partly-adequate	partly-suitable	unacceptable
unacceptable	partly-adequate	suitable	unacceptable
unacceptable	partly-adequate	very suitable	unacceptable
unacceptable	adequate	unsuitable	unacceptable
unacceptable	adequate	partly-suitable	unacceptable
unacceptable	adequate	suitable	unacceptable
unacceptable	adequate	very suitable	unacceptable
acceptable	inadequate	unsuitable	unacceptable
acceptable	inadequate	partly-suitable	unacceptable
acceptable	inadequate	suitable	unacceptable
acceptable	inadequate	very suitable	unacceptable
acceptable	partly-adequate	unsuitable	unacceptable
acceptable	partly-adequate	partly-suitable	acceptable
acceptable	partly-adequate	suitable	acceptable
acceptable	partly-adequate	very suitable	acceptable
acceptable	adequate	unsuitable	acceptable
acceptable	adequate	partly-suitable	acceptable
acceptable	adequate	suitable	acceptable
acceptable	adequate	very suitable	good
good	inadequate	unsuitable	unacceptable
good	inadequate	partly-suitable	acceptable
good	inadequate	suitable	acceptable
good	inadequate	very suitable	good
good	partly-adequate	unsuitable	acceptable
good	partly-adequate	partly-suitable	good
good	partly-adequate	suitable	good
good	partly-adequate	very suitable	very good
good	adequate	unsuitable	acceptable
good	adequate	partly-suitable	good
good	adequate	suitable	good
good	adequate	very suitable	very good
very good	inadequate	unsuitable	unacceptable
very good	inadequate	partly-suitable	acceptable
very good	inadequate	suitable	acceptable
very good	inadequate	very suitable	good
very good	partly-adequate	unsuitable	unacceptable
very good	partly-adequate	partly-suitable	good
very good	partly-adequate	suitable	good
very good	partly-adequate	very suitable	very good
very good	adequate	unsuitable	acceptable
very good	adequate	partly-suitable	good
very good	adequate	suitable	very good
very good	adequate	very suitable	excellent
excellent	inadequate	unsuitable	unacceptable
excellent	inadequate	partly-suitable	acceptable
excellent	inadequate	suitable	good
excellent	inadequate	very suitable	good
excellent	partly-adequate	unsuitable	acceptable
excellent	partly-adequate	partly-suitable	good
excellent	partly-adequate	suitable	good
excellent	partly-adequate	very suitable	very good
excellent	adequate	unsuitable	acceptable
excellent	adequate	partly-suitable	very good
excellent	adequate	suitable	excellent
excellent	adequate	very suitable	excellent

$$f_{\text{N Life Event}}(y) = \begin{cases} 1 & \text{if } y \text{ is unacceptable} \\ 2 & \text{if } y \text{ is acceptable} \\ 3 & \text{if } y \text{ is good} \\ 4 & \text{if } y \text{ is very good} \\ 5 & \text{if } y \text{ is excellent} \end{cases} . \tag{7.22}$$

Aggregation function for Life Event attribute in life event evaluation is presented in Table 7.7, which aggregates all combinations of *Maturity*, *Usability of LE* and *LE clarity* values into the output value of *Life Event*.

The *Function type 4* for attribute *LE Sophistication* aggregates numeric values from $D_{\text{AVG E-service}}$ into qualitative values in $D_{\text{LE Sophistication}}$, according to Table 7.5:

$$NQ_{\text{LE Sophistication}}(y) = \begin{cases} \text{unacceptable} & \text{if } y \in [1, 1.7] \\ \text{acceptable} & \text{if } y \in (1.7, 2.5] \\ \text{good} & \text{if } y \in (2.5, 3.3] \\ \text{very good} & \text{if } y \in (3.3, 4.1] \\ \text{excellent} & \text{if } y \in (4.1, 5] \end{cases} . \tag{7.23}$$

N E-service's aggregation function transforms qualitative values into numeric integer values. The function is a *Function type 2*. Its purpose is to prepare the evaluations of e-services for their averaging in the relational attribute *AVG E-service*. It aggregates qualitative values from $D_{\text{E-service}}$ into domain $D_{\text{N E-service}}$. It is defined as:

$$f_{\text{N E-service}}(y) = \begin{cases} 1 & \text{if } y \text{ is unacceptable} \\ 2 & \text{if } y \text{ is acceptable} \\ 3 & \text{if } y \text{ is good} \\ 4 & \text{if } y \text{ is very good} \\ 5 & \text{if } y \text{ is excellent} \end{cases} . \tag{7.24}$$

Similarly, the aggregation function for E-service attribute in e-service evaluation presented in Table 7.8 aggregates all combinations of *Clarity of e-Service*, *Information* and *Sophistication* values into the output value of *E-service*.

7.3.1 Relational Aggregation Functions

The overall model for e-portal evaluation uses two relational aggregation functions. The first function is defined in the model for evaluation of portals, for relational attribute *AVG Life Event*. The attributes counterpart attribute from model for assessment of life events is the root attribute *N Life Event*. The relational aggregation function is presented in Algorithm 7.1. The aggregation function outputs an average value of all numerical evaluations of life events. If there are no life events corresponding to the particular portal, the function raises an error, and the evaluation ends. This seems an abrupt ending to the evaluation procedure. However, consider the case where there would be no relational alternatives — in such cases the behavior would be undefined, because of the division by zero. Hence, the DM and/or the user of the model must guarantee that at least one relational alternative is defined for every portal.

The second relational aggregation function appears in the evaluation of life events, in the relational attribute *AVG E-service*. The attributes counterpart input attributes are *N E-service* and *E-service Type*, from the model for evaluating e-services. The relational

Table 7.7: The aggregation function for Life Event evaluation from Maturity, Usability of LE and LE Clarity.

Maturity	*Usability of LE*	*LE clarity*	*Life Event*
unacceptable	unsuitable	inadequate	unacceptable
unacceptable	unsuitable	partly-adequate	unacceptable
unacceptable	unsuitable	adequate	unacceptable
unacceptable	partly suitable	inadequate	unacceptable
unacceptable	partly suitable	partly-adequate	unacceptable
unacceptable	partly suitable	adequate	unacceptable
unacceptable	suitable	inadequate	unacceptable
unacceptable	suitable	partly-adequate	unacceptable
unacceptable	suitable	adequate	unacceptable
unacceptable	very suitable	inadequate	unacceptable
unacceptable	very suitable	partly-adequate	unacceptable
unacceptable	very suitable	adequate	unacceptable
acceptable	unsuitable	inadequate	unacceptable
acceptable	unsuitable	partly-adequate	unacceptable
acceptable	unsuitable	adequate	unacceptable
acceptable	partly suitable	inadequate	acceptable
acceptable	partly suitable	partly-adequate	acceptable
acceptable	partly suitable	adequate	acceptable
acceptable	suitable	inadequate	acceptable
acceptable	suitable	partly-adequate	acceptable
acceptable	suitable	adequate	acceptable
acceptable	very suitable	inadequate	acceptable
acceptable	very suitable	partly-adequate	acceptable
acceptable	very suitable	adequate	good
good	unsuitable	inadequate	unacceptable
good	unsuitable	partly-adequate	acceptable
good	unsuitable	adequate	acceptable
good	partly suitable	inadequate	acceptable
good	partly suitable	partly-adequate	good
good	partly suitable	adequate	good
good	suitable	inadequate	acceptable
good	suitable	partly-adequate	good
good	suitable	adequate	good
good	very suitable	inadequate	acceptable
good	very suitable	partly-adequate	good
good	very suitable	adequate	very good
very good	unsuitable	inadequate	unacceptable
very good	unsuitable	partly-adequate	acceptable
very good	unsuitable	adequate	acceptable
very good	partly suitable	inadequate	acceptable
very good	partly suitable	partly-adequate	good
very good	partly suitable	adequate	good
very good	suitable	inadequate	acceptable
very good	suitable	partly-adequate	good
very good	suitable	adequate	very good
very good	very suitable	inadequate	acceptable
very good	very suitable	partly-adequate	very good
very good	very suitable	adequate	very good
excellent	unsuitable	inadequate	unacceptable
excellent	unsuitable	partly-adequate	acceptable
excellent	unsuitable	adequate	acceptable
excellent	partly suitable	inadequate	acceptable
excellent	partly suitable	partly-adequate	good
excellent	partly suitable	adequate	good
excellent	suitable	inadequate	acceptable
excellent	suitable	partly-adequate	very good
excellent	suitable	adequate	excellent
excellent	very suitable	inadequate	good
excellent	very suitable	partly-adequate	very good
excellent	very suitable	adequate	excellent

Table 7.8: The aggregation function for E-service evaluation from Clarity of e-Service, Information and Sophistication.

Clarity of e-Service	*Information*	*Sophistication*	*E-service*
inadequate	inadequate	unsuitable	unacceptable
inadequate	inadequate	partly suitable	unacceptable
inadequate	inadequate	suitable	unacceptable
inadequate	inadequate	very suitable	good
inadequate	partly-adequate	unsuitable	unacceptable
inadequate	partly-adequate	partly suitable	unacceptable
inadequate	partly-adequate	suitable	acceptable
inadequate	partly-adequate	very suitable	good
inadequate	adequate	unsuitable	unacceptable
inadequate	adequate	partly suitable	unacceptable
inadequate	adequate	suitable	acceptable
inadequate	adequate	very suitable	good
partly-adequate	inadequate	unsuitable	unacceptable
partly-adequate	inadequate	partly suitable	unacceptable
partly-adequate	inadequate	suitable	acceptable
partly-adequate	inadequate	very suitable	good
partly-adequate	partly-adequate	unsuitable	acceptable
partly-adequate	partly-adequate	partly suitable	acceptable
partly-adequate	partly-adequate	suitable	good
partly-adequate	partly-adequate	very suitable	very good
partly-adequate	adequate	unsuitable	acceptable
partly-adequate	adequate	partly suitable	acceptable
partly-adequate	adequate	suitable	very good
partly-adequate	adequate	very suitable	excellent
adequate	inadequate	unsuitable	unacceptable
adequate	inadequate	partly suitable	unacceptable
adequate	inadequate	suitable	acceptable
adequate	inadequate	very suitable	good
adequate	partly-adequate	unsuitable	acceptable
adequate	partly-adequate	partly suitable	acceptable
adequate	partly-adequate	suitable	very good
adequate	partly-adequate	very suitable	excellent
adequate	adequate	unsuitable	acceptable
adequate	adequate	partly suitable	good
adequate	adequate	suitable	excellent
adequate	adequate	very suitable	excellent

aggregation function is presented in Algorithm 7.2, and is modeled according to Equation (7.2). This aggregation function also returns an error, in the case where there are no relational alternatives. The user and/or the DM need to guarantee that relational alternatives are supplied correctly, before the evaluation.

7.4 Evaluation of Alternatives

All alternatives were evaluated against the model constructed with DEXx. The original alternatives were provided by the authors Leben et al. (2006). In the study, the authors gathered alternative information in the time interval, from August 2002 to June 2005. In this thesis we present the alternatives and results retrieved in July 2003. The comparison of results is discussed later in Section 7.5.

The dependencies of the original alternatives are given in Table 7.9. There were 16 alternatives (representing portals), 65 relational alternatives (representing different dependent life events) and another 317 relational alternatives (representing different dependent e-services). In the table, 16 names of the portals are given in the left column. The middle column of the table gives the names of the life events supported by the corresponding portal. The right-most column of the table presents the number of e-services supporting the

Algorithm 7.1: Algorithm for relational aggregation of relational attribute *AVG Life Event*. It aggregates n numeric values into average of these values. Note, if there are no relational alternatives (life events corresponding to the portal), an error is returned.

Input: Values $V = (v_1, v_2, \ldots, v_n), v_i \in D_{\text{N Life Event}}$
Output: Aggregated value $avg \in D_{\text{AVG Life Event}}$

```
1  N = 0;
2  sum = 0;
3  for v_i ∈ V do
4  |   N = N + 1;
5  |   sum = sum + v_i;
6  end
7  if N == 0 then                          /* No relational alternatives */
8  |   return error;
9  else
10 |   avg = sum/N;
11 |   return avg;
```

life event to its left. For example, the portal "SLO-Slovenia-eUprava" has four supported life-events: "SLO-company", "SLO-driving-license", "SLO-passport" and "SLO-relocation". The number of e-services supporting the life events are 3, 3, 3 and 2, respectively.

The text behind the dash in the life event column represents the actual life events supported by a corresponding portal. The possible life events are:

- "company" represents a life event of creating a company,
- "driving-license" is a life event of acquiring a driving license,
- "employment" is a life event of acquiring correct documentation for employment,
- "marriage" represents a life event of acquiring documentation for marriage,
- "passport" represents a life event of acquiring a passport,
- "relocation" is a life event of changing the address of a person, and
- "tax" represents a life event of submitting a personal tax report.

Evaluation of the supplied alternatives was done on the model described in Section 7.2, aggregation functions described in Section 7.3, relational aggregation functions described in Section 7.3.1, and algorithm (*evaluate*) for evaluation of relational models described in Algorithm 4.2.

To compute the evaluations of each portal, the values of the inputs for the alternative were entered into the corresponding input attributes and the relational evaluation was done using algorithm *evaluate*.

According to Algorithm 4.2, the attributes of model *Portal* are topologically sorted, given the model structure. When the evaluation procedure encounters an aggregated attribute, it evaluates the corresponding aggregation function. While, when it encounters a relational attribute (*LE Solution Methods*) it gets all corresponding relational alternatives and evaluates them, using the model *Life Event*, and aggregates the evaluations with the relational function described in Algorithm 7.1. Finally, it sets the value of attribute *LE*

Algorithm 7.2: Algorithm for relational aggregation of relational attribute *AVG E-service*. It aggregates n numeric evaluations of E-services (*N E-service*) and their types (*E-service Type*) into weighted average of these values. Note, if there are no relational alternatives (e-services corresponding to the life event), an error is returned.

Input: Values $V = (v_1, v_2, \ldots, v_n), v_i \in D_{\text{N E-service}}$,
Values $T = (t_1, t_2, \ldots, t_n), t_i \in D_{\text{E-service Type}}$
Output: Aggregated value $avg \in D_{\text{AVG E-service}}$

```
1  vital_SUM = 0;
2  supplementary_SUM = 0;
3  vital_N = 0;
4  supplementary_N = 0;
5  avg = 0;
6  for i = 1; i ≤ n; i = i + 1 do
7  |  evaluation = V[i];
8  |  type = T[i];
9  |  if type == vital then
10 |  |  vital_SUM = vital_SUM + evaluation;
11 |  |  vital_N = vital_N + 1;
12 |  else
13 |  |  supplementary_SUM = supplementary_SUM + evaluation;
14 |  |  supplementary_N = supplementary_N + 1;
15 end
16 if vital_N + supplementary_N == 0 then      /* No relational alternatives */
17 |  return error;
18 else if supplementary_N == 0 then
19 |  VS = vital_SUM / vital_N;
20 |  avg = VS;
21 |  return avg;
22 else if vital_N == 0 then
23 |  SS = supplementary_SUM / supplementary_N;
24 |  avg = 2/3 + 1/3 SS;
25 |  return avg;
26 else
27 |  VS = vital_SUM / vital_N;
28 |  SS = supplementary_SUM / supplementary_N;
29 |  avg = 2/3 VS + 1/3 SS;
30 |  return avg;
```

Solution Methods to the computed value. Similarly, when evaluating each life event, when the algorithm gets to the relational attribute *LE Sophistication*, it evaluates all corresponding e-services on model for E-service assessment, and computes the relational aggregation function on their evaluations, according to Algorithm 7.2.

7.5 Comparison of Results

The computed results were compared to the results computed in the original study by the authors Leben et al. (2006). Evaluations of portals computed with DEXx are presented in Figure 7.2. On the horizontal axis all evaluated portals are displayed, and on the vertical

axis the evaluation of each corresponding portal. The evaluations are taken from the scale of *Portal* attribute D_{Portal}. We must stress that all evaluations computed with DEXx implementation are identical to the evaluations computed by the original study.

Furthermore, we must point out that there was far less manual work (none) included in the evaluation with DEXx. This is due to the fact that DEXx supports native relational aggregation, which eliminates manual intervention during evaluation and relational aggregation. Recall that the authors Leben et al. (2006) were required to employ Microsoft Excel twice during relational aggregation. There was a need to manually transform qualitative values into numeric values using Equation (7.1), aggregate values as averages or according to Equation (7.2), and finally map the computed numeric values to qualitative values according to Tables 7.4 and 7.5. Even more, formalizing the relational aggregation (and computing the relational aggregation functions with a computer) eliminated the risk of human error — while carrying out the manual aggregation.

With DEXx, all manual intervention was eliminated, because all previously manually managed aggregations were now done automatically, with correctly defined procedures and aggregation functions. For the correct translation of values, we employed two new attributes (N E-service and N Life Event), with new aggregation functions (*Function type 2*), which transform qualitative values into numeric integer values (see Equations (7.22) and (7.24)). Furthermore, for relational aggregation, we defined two additional numeric attributes: AVG Life Event and AVG E-service. Their corresponding relational aggregation functions are provided in Algorithms 7.1 and 7.2. Finally, to transform the numeric results, we implemented two additional aggregation functions (*Function type 4*), presented in Equations (7.21) and (7.23). All other parts of the model (attributes, model structure and aggregation functions) were used directly from the model acquired from the authors Leben et al. (2006).

The implementation of the models developed during the original study using DEXx resulted in computing the same evaluations, as in the original study. With that, we have shown that only minor modifications were needed on the original model, to generate the same results, eliminating the need for manual intervention and aggregation.

The evaluations of all life events corresponding to portals are presented in Figure 7.3. The horizontal axis shows all evaluated portals and the vertical axis the achieved evaluation of the life event, from the scale $D_{\text{Life Event}}$. Each portal has one bar for each of the life events that particular portal supports. For example, portal "SLO" has four bars, for each of the 4 supported life events. Furthermore, each life event type is assigned a unique color: evaluation of "company" life event is red, "driving-license" is blue, "employment" is green, "marriage" is cyan, "passport" is purple, "relocation" is yellow and evaluation of "tax" is displayed in orange.

The plot shown in Figure 7.3 also gives some indication on what the final portal evaluation will be. For example, all life events for "SLO-Slovenia-eUprava" portal are evaluated as *unacceptable*. Consequently, the overall evaluation of "eUprava" portal is *unacceptable*. Similarly, the life event evaluations for "ESP" indicate an *unacceptable* portal evaluation. This feature shows that in the case of bad evaluations of life events, the portal evaluation cannot be good (even though other factors could be very good). Note however that better evaluations of life events do not necessarily mean better results in the final evaluations of portals.

7.6 Concluding Remarks

The study for evaluation of e-portals required treatment of relational alternatives and relational models. The purpose of use-case presentation was to repeat the study, by im-

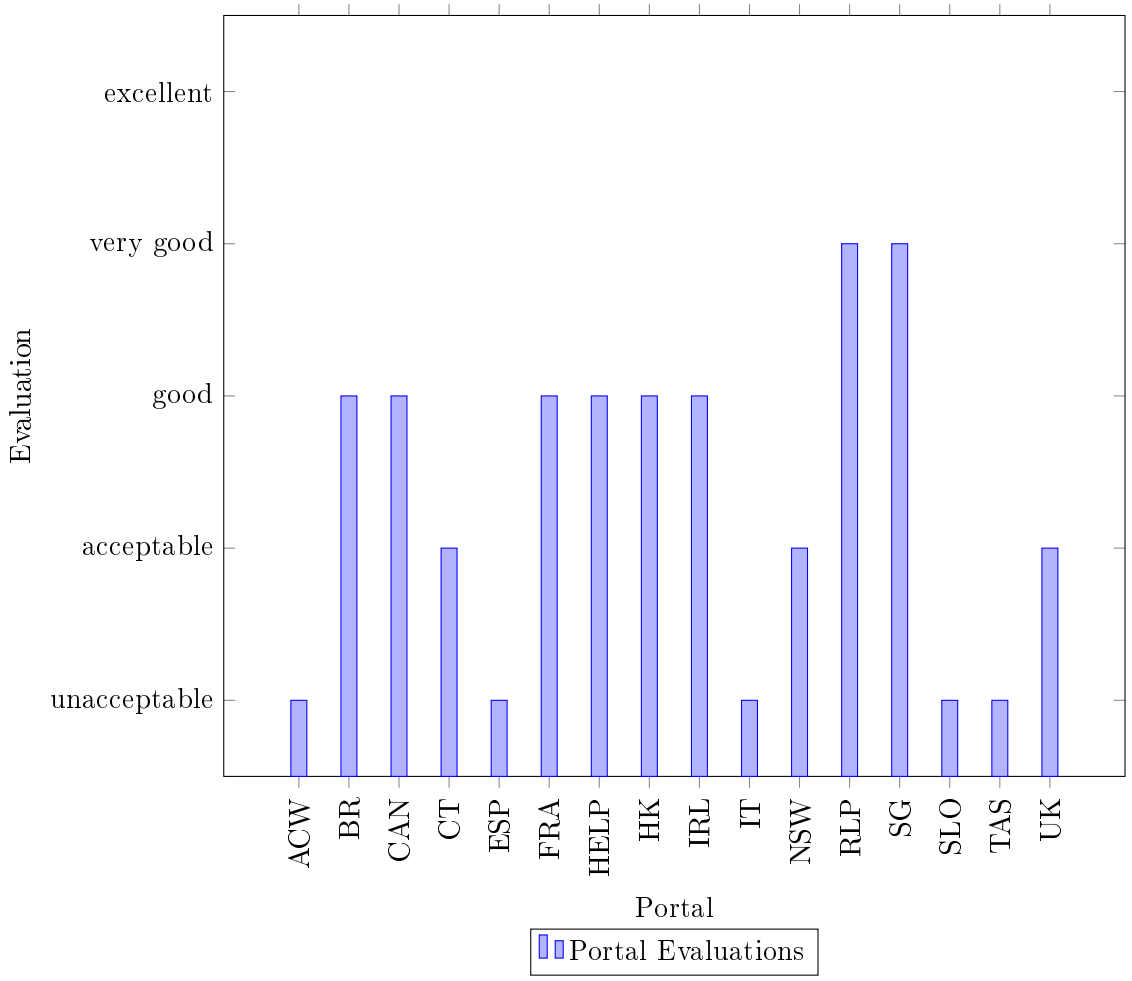

Figure 7.2: Comparison of final evaluations of e-Portals implemented using DEXx. The blue bars represent evaluations computed by the library, based on the input values of alternatives.

plementing the models using DEXx, with DEXx implementation. The newly implemented case study uses native support for relational models and relational alternatives, with small usage of numeric attributes. The results produced using DEXx show that study of e-portals can be indeed implemented with DEXx library, producing identical results as the original study, and without any manual work during evaluation of alternatives. Even more, the risk of errors during relational aggregations is reduced, in comparison with the original study.

For the criteria determined in methodology, we can assess this use-case as follows:

- The implementation of the e-portal model is possible, using the DEXx framework.
- The implementation of e-portals was previously done by evaluating DEX models in DEXi, and performing relational aggregation in Microsoft Excel. In the new implementation of the e-portals model, the numeric attributes and the relational models extension is used. Without the latter, the implementation of the model would not be possible.
- The results of evaluations received from the authors of the original study, and results computed with the DEXx model implementation are identical.

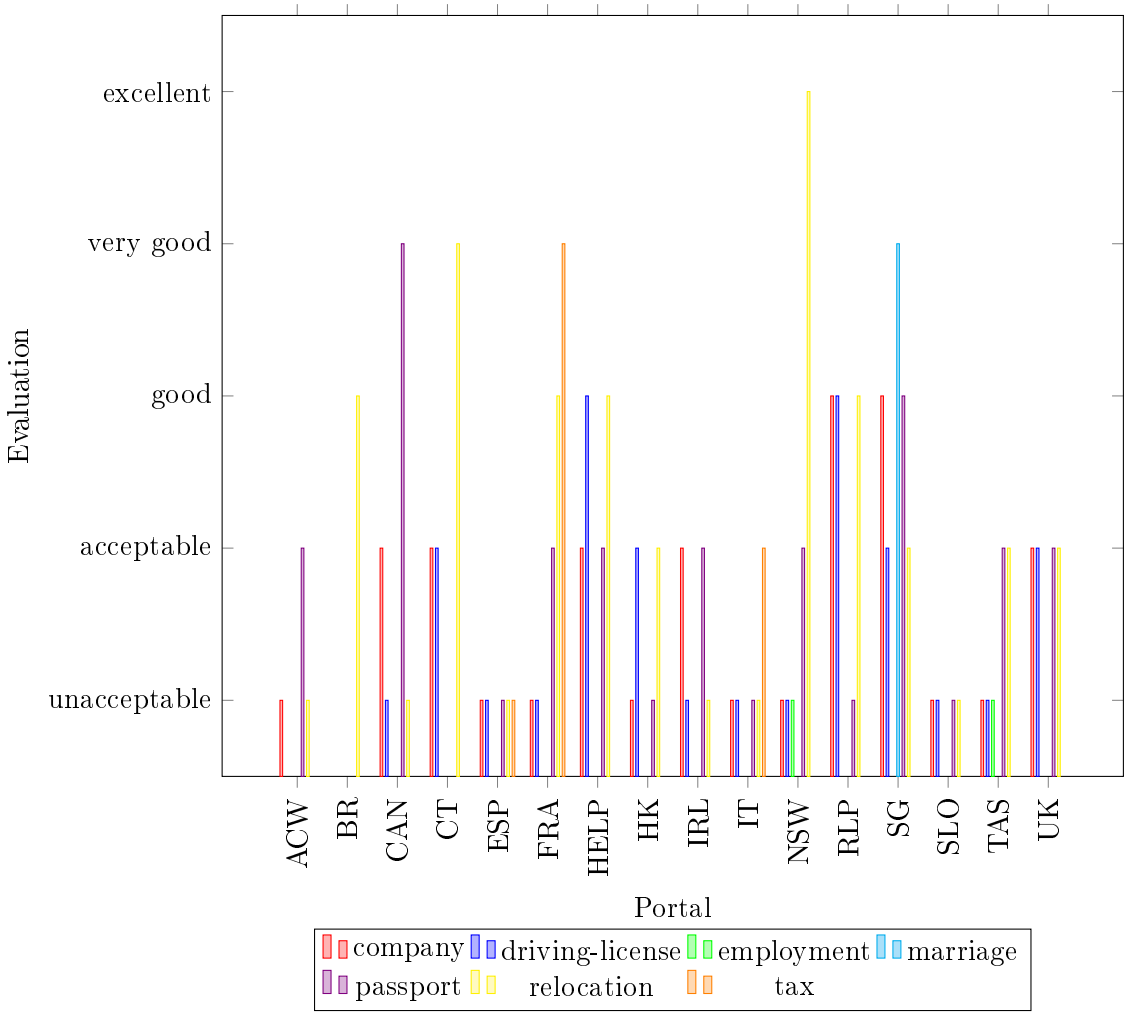

Figure 7.3: Results for different life event evaluations. The considered life events are company, driving-license, employment, marriage, passport, relocation and tax. The evaluations are split according to the portal in which they appear. If the portal does not have a particular life event, then there is no bar in corresponding color.

- Evaluation time is not known for the original study, hence it cannot be compared. However, we can assume the manual relational aggregation time using Microsoft Excel takes considerably more time than using the natively implemented relational aggregation functions in DEXx.

  We can safely assume that evaluation time of alternatives in the model is decreased, in comparison with previous evaluation time.

Table 7.9: The portals considered in the study are displayed in the left column. The middle column shows the life events supported by the corresponding portal, and on the right is the number of e-services supporting the particular life event.

Portal	Life Event	Number of e-Services
ACW-Commonwealth	ACW-company	2
	ACW-passport	2
	ACW-relocation	1
BR-Bremen	BR-relocation	9
CAN-Canada	CAN-company	18
	CAN-driving-license	1
	CAN-passport	1
	CAN-relocation	3
CT-Central Territory	CT-company	3
	CT-driving-license	4
	CT-relocation	3
ESP-Spain	ESP-company	3
	ESP-driving-license	9
	ESP-passport	3
	ESP-relocation	6
	ESP-tax	2
FRA-France	FRA-company	1
	FRA-driving-license	8
	FRA-passport	5
	FRA-relocation	11
	FRA-tax	1
HELP-Austria	HELP-company	9
	HELP-driving-license	8
	HELP-passport	10
	HELP-relocation	17
HK-Hong Kong	HK-company	5
	HK-driving-license	5
	HK-passport	1
	HK-relocation	3
IRL-Ireland	IRL-company	8
	IRL-driving-license	4
	IRL-passport	3
	IRL-relocation	1
IT-Italy	IT-company	2
	IT-driving-license	7
	IT-passport	3
	IT-relocation	2
	IT-tax	2
NSW-New South Wales	NSW-company	3
	NSW-driving-license	1
	NSW-employment	1
	NSW-passport	2
	NSW-relocation	1
RLP-Rheinland-Pfalz-Lotse	RLP-company	7
	RLP-driving-license	5
	RLP-passport	6
	RLP-relocation	4
SG-Singapore	SG-company	7
	SG-driving-license	5
	SG-marriage	4
	SG-passport	2
	SG-relocation	3
SLO-Slovenia-eUprava	SLO-company	3
	SLO-driving-license	3
	SLO-passport	3
	SLO-relocation	2
TAS-Tasmania	TAS-company	3
	TAS-driving-license	13
	TAS-employment	8
	TAS-passport	2
	TAS-relocation	6
UK-United Kingdom	UK-company	9
	UK-driving-license	12
	UK-passport	1
	UK-relocation	15

Chapter 8

Model for Assessment of Bank Reputational Risk

The decision support model for the assessment of bank reputational risk (Bohanec et al., 2013, 2014) was developed in the scope of European Project FIRST (Project FIRST, 2013). The model's innovative aspect is the use of diverse data of different types and diverse sources. Data handled by the model is structured and unstructured, qualitative and numeric, and internal and external. The model was also used in a setting with processing large amounts of data.

Reputational risk is defined as "the risk arising from negative perception on the part of customers, counterparties, shareholders, investors, debt-holders, market analysts, other relevant parties or regulators that can adversely affect bank's ability to maintain existing, or establish new business relationships and continued access to sources of funding" (Bohanec et al., 2013; Project FIRST, 2013). The *reputational risk model* (RIM) is aimed at estimating bank reputational risk as an indicator supporting risk managers. The model takes input from readily available structured bank data, and from supplementary unstructured external data extracted from various web resources. This external unstructured data is presented as text from different blogs, web-pages, online newspapers and financial documents. This data is then presented to the model in the form of a *sentiment* towards some entity (be it bank, stock option, etc.). Specifically, sentiment is a positive or negative view, attitude, emotion or appraisal on the studied object from a document author or actor (Bohanec et al., 2013; Project FIRST, 2013). The retrieval of unstructured text documents, its aggregation and presentation was out of the scope of the studies (Bohanec et al., 2013, 2014) and also out of scope of this thesis. However, more information on these subjects can be read on the dissemination materials of the European Project FIRST (Project FIRST, 2013).

Usually, a bank models their data according to the hierarchical scheme as follows: some given *customer* buys some given *financial product*. The financial products are produced by *counterparts*. *Banks* provide different products produced by the counterparts. These relationships are modeled in a "one-to-many" way. One product can be bought by many customers, one counterpart can produce many products, and one bank can offer many counterparts. Hence, one would expect that relational models are applicable in the context of bank reputational risk assessment.

The purpose of the use-case presentation is to reimplement the model from the original study, natively in the DEXx framework. The newly implemented model uses relational models and relational aggregation functions, numeric attributes and newly introduced primitive function types.

To evaluate a bank for reputational risk, all supported counterparts need to be assessed.

To evaluate each counterpart, all produced products need to be evaluated. And finally, to get a reputational risk of a product, all product/customer pairs need to be evaluated. With that in mind, a model for bank reputational risk assessment was developed. The model is in essence a relational model, which is split into four parts:

1. Model BA for *Bank reputational risk assessment* uses data from evaluations of corresponding counterparts, and computes a weighted average over all counterparts. The value presents the final bank's reputational risk.

2. Model CP for *Counterpart reputational risk assessment* is used for evaluation of given counterpart's reputational risk. The model uses relational inputs from model for evaluation of reputational risk of a product. It aggregates all corresponding produced products as weighted average of their evaluations. It also supplies the model for bank reputational risk, with the *weight* of the given counterpart. Weight of a counterpart is computed as the sum of corresponding product's weights.

3. Model PR for *Product reputational risk assessment* computes the risk for a given product, based on the product/customer pairs for all customers that hold this particular product. It computes the reputational risk by using the average and the weighted average of corresponding product/customer evaluations. Weights of a given product are also outputted, computing them with weighted averages of relative product volume and relative number of customers, holding this product.

4. Model PC for *Product/Customer reputational risk assessment* is the only model that does not have any relational inputs. This is also the only model in the study which uses qualitative attributes. The model computes the reputational risk of a given product/customer combination. It considers the counterpart's qualitative sentiment, qualitative performance assessment of the product, qualitative mismatching assessment of the product, and qualitative relative customers' volumes.

Models for bank, counterpart, product and product/customer pair assessment are relationally structured as presented in Figure 8.1. The figure shows that multiple evaluations of product/customer pairs influence a single product assessment (through relational attributes rx_P and counterpart relational inputs ox_{PC}). Further, multiple evaluations of products influence a single counterpart, through relational attributes rx_C and counterpart input attributes ox_P. Again, bank reputational risk assessment if assessed only on the evaluations of the respective counterparts, through attribute rx_B and counterpart relational inputs ox_C.

The model is adapted to be used on time-series data — the purpose is to use the model daily. The bank tracks its customers, products and counterparts on an even smaller time interval, however, the sentiment is provided only at daily intervals. The Figure 8.2 (Bohanec et al., 2013, 2014; Project FIRST, 2013) shows a scheme on how the model can be used on the time-series data. When the data from the bank and sentiment engine is available, the model can be evaluated. The results combined from multiple points in time construct a time-series of bank's reputational risk assessment through time. Furthermore, not only the final evaluations are of interest, every value assigned to some attribute also constructs a specific time-series, which influences the end result.

The result of a single bank evaluation yields a numeric result from the interval $[1, 5]$, where a higher number represents a higher risk. However, if the evaluations are done on time-series data, a time-series result is obtained. Even more, for each of the internal variables, the corresponding time-series can be obtained to further understand the end evaluations.

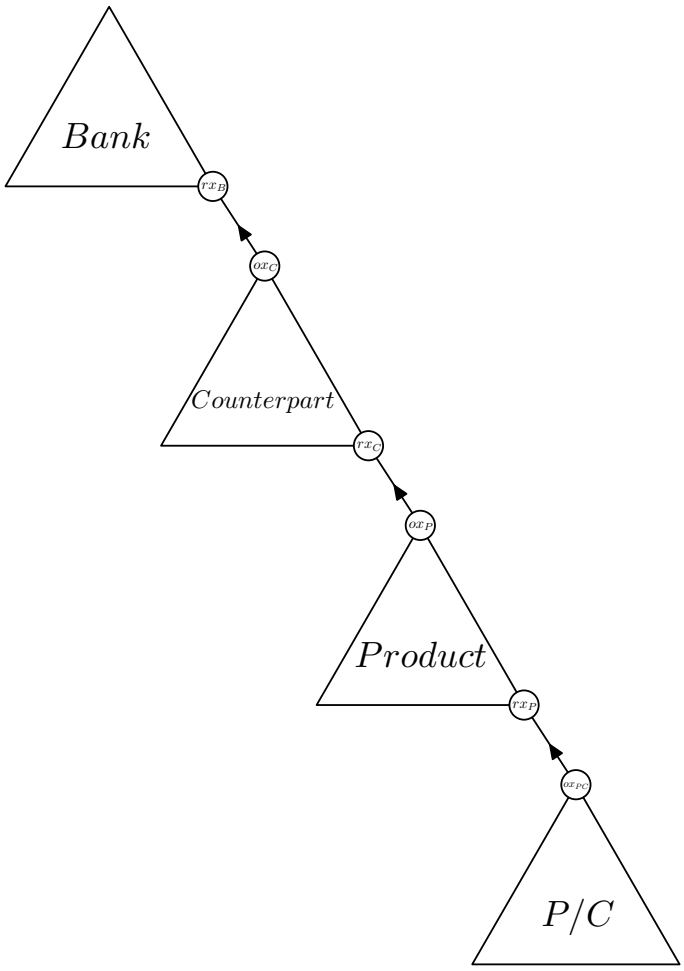

Figure 8.1: The configuration of main model for bank reputational risk assessment with the relationally connected model for counterpart assessment, which in turn depends on respective product assessments, and each product assessment depends on multiple product/customer (P/C) pair assessments.

The model as a whole has eight regular input attributes that are supplied through alternatives. The lowest level input attributes are in the *PC* model. All the following four attributes are firstly discretized (in a preprocessing step), before they are entered into alternative values:

- qS: *qualitative sentiment* assessment, which is a discretized counterpart *sentiment indicator*. The value is assigned based on the Equation (8.1) which discretizes value S, which in turn is computed according to Equation (8.2). Here, $S_{SP} \in [0,1]$ presents the *short-period sentiment indicator* for the corresponding counterpart and $S_{LP} \in [0,1]$ presents the *long-period (90 day) average sentiment indicator* for the corresponding counterpart.

$$qS(S) = \begin{cases} \text{very negative} & \text{if } S \leq -30 \\ \text{high negative} & \text{if } S \in (-30, -20] \\ \text{medium negative} & \text{if } S \in (-20, -10] \\ \text{low negative} & \text{if } S \in (-10, -1] \\ \text{neutral} & \text{if } S > -1 \end{cases} \tag{8.1}$$

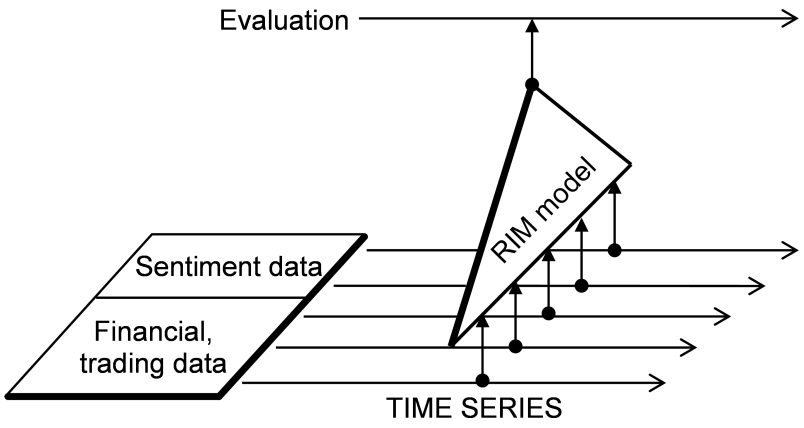

Figure 8.2: The model for bank's reputational risk assessment can be used on time-series data. For each point in time, when the data is available (from the sentiment and bank), the model can be run and results viewed as time-series (Bohanec, Aprile, Constante, Foti, and Trdin, 2013, 2014; Project FIRST, 2013).

$$S = 0.7 \cdot S_{SP} + 0.3 \cdot S_{LP} \tag{8.2}$$

- qP: *qualitative performance* assessment of the product for a given customer. A numeric value for a specific product is computed using Equation (8.3), where PP is the absolute *product performance* computed from the point when the customer bought this product, and BP is the *benchmark performance* of similar products, from the point when the customer bought this product. Furthermore, value P is discretized according to Equation (8.4), before it is considered as an input value.

$$P = PP + 0.1 \cdot (PP - BP) \tag{8.3}$$

$$qP(P) = \begin{cases} \text{very high} & \text{if } P \leq -100 \\ \text{high} & \text{if } P \in (-100, -50] \\ \text{medium} & \text{if } P \in (-50, -25] \\ \text{low} & \text{if } P \in (-25, -10] \\ \text{in-line} & \text{if } P > -10 \end{cases} \tag{8.4}$$

- qM: *qualitative mismatching* assessment of the customer. The value is acquired through evaluation of Equation (8.5), where SRI is a *synthetic risk indicator* provided by the bank, and RP is the customer's *risk profile*, which is provided by the customer in a questionnaire. Finally, the value is discretized according to Equation (8.6).

$$M = SRI - RP \tag{8.5}$$

$$qM(M) = \begin{cases} \text{very high} & \text{if } M \geq 3.5 \\ \text{high} & \text{if } M \in [2.5, 3.5) \\ \text{medium} & \text{if } M \in [1.5, 2.5) \\ \text{low} & \text{if } M \in [0.5, 1.5) \\ \text{in-line} & \text{if } M < 0.5 \end{cases} \tag{8.6}$$

- qRV_{1c}: is a qualitative value of *relative customer's volume* of a given *product*. The value is computed by using Equation (8.7), where V_1 is the *volume* of this particular *product* for this *customer* and V_C are the *total assets* of this *customer*. The numeric value of RV_{1c} is discretized using Equation (8.8).

$$RV_{1c} = \frac{V_1}{V_C} \tag{8.7}$$

$$qRV_{1c}(RV_{1c}) = \begin{cases} \text{very high} & \text{if } RV_{1c} \geq 30 \\ \text{high} & \text{if } RV_{1c} \in [20, 30) \\ \text{medium} & \text{if } RV_{1c} \in [10, 20) \\ \text{low} & \text{if } RV_{1c} \in [5, 10) \\ \text{in-line} & \text{if } RV_{1c} < 5 \end{cases} \tag{8.8}$$

Four additional inputs are provided as alternatives to model PR:

- RNp: is a numeric value representing the *relative number of customers* holding a given *product*. The value is computed by using Equation (8.9), where Np is the *total number of customers* holding the *product*, and TN is the *total number of customers* of the *bank*.

$$RNp = \frac{Np}{TN} \tag{8.9}$$

- RVp: is a numeric value representing the *relative number of given product's volumes* in a *bank*. The value is computed by using Equation (8.10), where Vp is the *bank's volume* of given *product*, and TA are the *total assets* of the *bank*.

$$RVp = \frac{Vp}{TA} \tag{8.10}$$

- $SRNp$: is a numeric value representing the *sum of relative number of customers* holding a *product*, over a specified set of *products*. Given a set of products $p \in P$, the value is computed as per Equation (8.11). Note that this value is computed only once for the evaluation of the whole model.

$$SRNp = \sum_{p \in P} RNp \tag{8.11}$$

- $SRVp$: is a numeric value representing the *sum of relative number of product volumes*, over a specified set of *products*. Given a set of products $p \in P$, the value is computed as per Equation (8.12). Note that this value is computed only once for the evaluation of the whole model.

$$SRVp = \sum_{p \in P} RVp \tag{8.12}$$

Note that computation of $SRNp$ and $SRVp$ indicate that the model can be used to evaluate just a *part* of the bank — here the part indicates a subset of products of interest. If a reputational risk of a whole bank is evaluated, then both $SRNp$ and $SRVp$ become 1.

8.1 Model Description and Structure

The model structure for Bank reputational risk assessment was constructed from the study (Bohanec et al., 2013, 2014; Project FIRST, 2013). The models BA, CP, PR and PC are presented in Tables 8.1, 8.2, 8.3 and 8.4, respectively.

On the left side of each table, the model structure is presented. The top-most attribute is the root of the model. Note that models CP (Table 8.2), PR (Table 8.3) and PC model (Table 8.4) each have two roots. The corresponding roots are RI_{CP} and w_{CP}, RI_{PR} and w_{PR}, and nRI_1 and nRV_1c. Each attribute's direct descendants are connected with a line and indented to the right. Attributes written in blue text are the relational attributes (RI_{BA} in model BA, $weighted\_RI_p$ and w_{CP} in model CP, and $weighted\_nRI_i$, $sum\_nV_iP_i$ and RI_u in model PR). The descendants of relational attributes are attributes from subordinate relational models. Descendant of RI_{BA} are RI_{CP} and w_{CP}, from the model CP. RI_{PR} and w_{PR} are descendants of $weighted\_RI_P$ and w_{CP} from the model PR. Descendants of $weighted\_nRI_i$, $sum\_nV_iP_i$ and RI_u are nRI_1 and nRV_1c, from the model PC. Attributes displayed in green text are the hierarchical attributes — attributes which influence multiple parts of the same model and subsequently have multiple parent attributes (see Section 4.1).

On the right side of each table, the scales for corresponding attributes are given. All scales are preferentially ordered in an increasing order. For numeric values, a greater value means greater preference. For the qualitative values, appearing in the model for product/customer assessment, the worst values are written in red, the best values are written in green, and the intermediate values are written in black text.

Formally, model for assessment of the bank reputational risk (BA) has the following attribute (the equations are derived from Table 8.1):

$$X = \{RI_{BA}\}. \tag{8.13}$$

The descendants of attributes are described by mapping S (see Equation (3.4)):

$$S(RI_{BA}) = \{\}. \tag{8.14}$$

Now, because the model for bank reputational risk assessment has only one relational attribute, which is also the root, there are no aggregated attributes:

$$aggAttributes_{\mathrm{BA}} = \{\}. \tag{8.15}$$

Furthermore, $modelInputs_{\mathrm{Bank}}$ can be established:

$$modelInputs_{\mathrm{BA}} = \{RI_{BA}\}. \tag{8.16}$$

Finally, the computed $modelOutputs_{\mathrm{Bank}}$ are:

$$modelOutputs_{\mathrm{BA}} = \{RI_{BA}\}. \tag{8.17}$$

The scales of the attributes are shown on the right side of the Table 8.1. The scale for the one attribute is:

$$D_{RI_{BA}} = [1, 5]. \tag{8.18}$$

Since the model BA has a subordinate relational model, its only attribute is also a relational attribute (rx). Its counterpart attributes (ox) from the relational model for counterpart assessment are RI_{CP} and w_{CP}, which respectively represent the counterpart's reputational risk assessment and the contributed weight of the counterpart.

Table 8.1: The model for reputational risk assessment of a bank (BA). On the left side, the structure of the attributes is presented, and on the right side, their corresponding numeric scales are displayed.

Attribute	**Scale**
RI_{BA}	$[1, 5]$
RI_{CP}	$[1, 5]$
w_{CP}	$[0, 1]$

Similarly, attributes X, successors S, aggregated attributes $aggAttributes_{\text{CP}}$, model inputs $modelInputs_{\text{CP}}$, model outputs $modelOutputs_{\text{CP}}$ and scales D, are constructed from Table 8.2 for the model for counterpart's reputational risk assessment (CP).

$$X = \{RI_{CP}, weighted\_RI_{PR}, w_{CP}\}, \tag{8.19}$$

$$\begin{gathered} S(RI_{CP}) = \{weighted\_RI_{PR}, w_{CP}\}, \\ S(weighted\_RI_{PR}) = \{\}, \\ S(w_{CP}) = \{\}, \end{gathered} \tag{8.20}$$

$$aggAttributes_{\text{CP}} = \{RI_{CP}\}, \tag{8.21}$$

$$modelInputs_{\text{CP}} = \{weighted\_RI_{PR}, w_{CP}\}, \tag{8.22}$$

$$modelOutputs_{\text{CP}} = \{RI_{CP}, w_{CP}\}, \tag{8.23}$$

$$\begin{gathered} D_{RI_{CP}} = [1, 5], \\ D_{weighted\_RI_{PR}} = [1, \infty), \\ D_{w_{CP}} = [0, 1]. \end{gathered} \tag{8.24}$$

The model CP has a subordinate relational model (model for assessment of product's reputational risk PR), so it also has two relational attributes (rx):

1. Attribute $weighted\_RI_{PR}$ has two counterpart relational inputs: assessment of product's reputational risk (RI_{PR}) and the respective weight of the product w_{PR}.

2. Attribute w_{CP} has one counterpart relational input attribute — the contributed weight of the respective product w_{PR}.

Table 8.2: The model for reputational risk assessment of a counterpart (CP). On the left side, the structure of the attributes is presented, and on the right side, their corresponding numeric scales are displayed.

Attribute	**Scale**
RI_{CP}	$[1, 5]$
$weighted\_RI_{PR}$	$[1, \infty)$
RI_{PR}	$[1, 5]$
w_{PR}	$[0, 1]$
w_{CP}	$[0, 1]$
w_{PR}	$[0, 1]$
w_{CP}	$[0, 1]$

Attributes X, successors S, aggregated attributes $aggAttributes_{\mathrm{PR}}$, model inputs $modelInputs_{\mathrm{PR}}$, model outputs $modelOutputs_{\mathrm{PR}}$ and scales D, are constructed from Table 8.3 for the model for product's reputational risk assessment (PR).

$$X = \{RI_{PR}, RI_w, weighted\_nRI_i, sum\_nV_iP_i, RI_u, w_P, RNp, RVp, SRNp, SRVp\}, \tag{8.25}$$

$$\begin{gathered} S(RI_{PR}) = \{RI_w, RI_u\}, \\ S(RI_w) = \{weighted\_nRI_i, sum\_nV_iP_i\}, \\ S(w_{PR}) = \{RNp, RVp, SRNp, SRVp\}, \\ S(weighted\_nRI_i) = \{\}, \\ S(sum\_nV_iP_i) = \{\}, \\ S(RI_u) = \{\}, \\ S(RNp) = \{\}, \\ S(RVp) = \{\}, \\ S(SRNp) = \{\}, \\ S(SRVp) = \{\}, \end{gathered} \tag{8.26}$$

$$aggAttributes_{\mathrm{PR}} = \{RI_{PR}, RI_w, w_{PR}\}, \tag{8.27}$$

$$modelInputs_{\mathrm{PR}} = \{weighted\_nRI_i, sum\_nV_iP_i, RI_u, RNp, RVp, SRNp, SRVp\}, \tag{8.28}$$

$$modelOutputs_{\mathrm{PR}} = \{RI_{PR}, w_{PR}\}, \tag{8.29}$$

$$\begin{gathered} D_{RI_{PR}} = [1, 5], \\ D_{RI_w} = [1, 5], \\ D_{weighted\_nRI_i} = [1, \infty), \\ \dots, \\ D_{SRVp} = [0, 1]. \end{gathered} \tag{8.30}$$

The model PR has a subordinate relational model (model for assessment of product/customer pair's reputational risk PC), subsequently also has three relational attributes (rx):

1. Attribute $weighted\_nRI_i$ has two counterpart relational inputs: numeric reputation risk of product/customer pair (nRI_1) and numeric relative customer's volume of the product (nRV_1c).

2. Attribute $sum\_nV_iP_i$ has one counterpart relational input attribute — the relative customer's volume of the product nRV_1c.

3. Attribute RI_u uses the counterpart attribute for numeric reputation risk of product/customer pair (nRI_1).

Table 8.3: The model for reputational risk assessment of a product (PR). On the left side, the structure of the attributes is presented, and on the right side, their corresponding numeric scales are displayed.

Attribute	**Scale**
RI_{PR}	$[1,5]$
RI_w	$[1,5]$
$weighted\_nRI_i$	$[1,\infty)$
nRI_1	$[1,5]$
nRV_1c	$[1,5]$
$sum\_nV_iP_i$	$[1,\infty)$
nRV_1c	$[1,5]$
RI_u	$[1,5]$
nRI_1	$[1,5]$
w_{PR}	$[0,1]$
RNp	$[0,1]$
RVp	$[0,1]$
$SRNp$	$[0,1]$
$SRVp$	$[0,1]$

Finally, the values for X, S, aggregated attributes $aggAttributes_{\text{PC}}$, $modelInputs_{\text{PC}}$, $modelOutputs_{\text{PC}}$ and scales D, can be constructed from Table 8.4 for the model for product/customer pair's reputational risk assessment (PC).

$$X = \{nRI_1, qRI_1, qS, qPM, qP, qM, qRV_1c, nRV_1c\}, \tag{8.31}$$

$$\begin{gathered} S(nRI_1) = \{qRI_1\}, \\ S(qRI_q) = \{qS, qPM, qRV_1c\}, \\ S(qPM) = \{qP, qM\}, \\ S(nRV_1c) = \{qRV_1c\}, \\ S(qS) = \{\}, \\ S(qP) = \{\}, \\ S(qM) = \{\}, \\ S(qRV_1c) = \{\}, \end{gathered} \tag{8.32}$$

$$aggAttributes_{\text{PC}} = \{nRI_1, qRI_1, qPM, nRV_1c\}, \tag{8.33}$$

$$modelInputs_{\text{PC}} = \{qS, qP, qM, qRV_1c\}, \tag{8.34}$$

$$modelOutputs_{\mathrm{PC}} = \{nRI_1, nRV_1c\}, \tag{8.35}$$

$$\begin{aligned} D_{nRI_1} &= [1,5], \\ D_{qRI_1} &= (\text{very high}, \text{high}, \text{medium}, \text{medium-low}, \text{low}), \\ &\dots, \\ D_{nRV_1c} &= [1,5], \\ D_{qRV_1v} &= (\text{very high}, \text{high}, \text{medium}, \text{medium-low}, \text{low}). \end{aligned} \tag{8.36}$$

Table 8.4: The model for qualitative assessment of reputational risk of product/customer pair (PC). On the left side, the structure of the attributes is presented, and on the right side, their corresponding scales are displayed. The scales are ordered from worst values (displayed in red) to best values (displayed in green). Additionally, two numeric scales are used, which are also preferentially ordered.

Attribute	Scale
nRI_1	$[1,5]$
qRI_1	very high, high, medium, medium-low, low
qS	very negative, high negative, medium-negative, low-negative, neutral
qPM	very high, high, medium, low, in-line
qP	very high, high, medium, low, in-line
qM	very high, high, medium, low, in-line
qRV_1c	very high, high, medium, medium-low, low
nRV_1c	$[1,5]$
qRV_1c	very high, high, medium, medium-low, low

The model for product/customer pair's reputational risk assessment does not have any subordinate relational models, hence there are no relational attributes and no relationally connected counterpart attributes.

8.2 Aggregation Functions

This section presents the aggregation functions for each of the developed models. All numeric aggregation functions (*Function type 3*), qualitative to numeric transformation functions (*Function type 2*), and one qualitative function (*Function type 1*) is presented. *Function type 1* is needed in the model PC. Later in Section 8.2.1 the used numeric relational aggregation functions are presented.

Model BA has only one relational aggregation function, which is covered in the following subsection.

Model CP has one aggregation function, which maps from values of weighted sum of corresponding products ($weighted\_RI_{PR}$) and sum of their weights (w_{CP}) to a counterpart's reputational risk (RI_{CP}). The function is:

$$f_{RI_{CP}}(weighted\_RI_{PR}, w_{CP}) = \frac{weighted\_RI_{PR}}{w_{CP}}. \tag{8.37}$$

The model PR has three functions of type 3. The first function maps from the reputational risk acquired from normalized weighted product/customer pair reputational risk RI_w and average product/customer pair's reputational risk RI_u to reputational risk of the product RI_{PR}:

$$f_{RI_{PR}}(RI_w, RI_u) = 0.6 \cdot RI_w + 0.4 \cdot RI_u. \tag{8.38}$$

The second numeric function maps from the weighted sum of the product/customer pair's reputational risk $weighted\_nRI_i$ and the sum of the weights $sum\_nV_iP_i$ to the normalized weighted product/customer pair reputational risk RI_w:

$$f_{RI_w}(weighted\_nRI_i, sum\_nV_iP_i) = \frac{weighted\_nRI_i}{sum\_nV_iP_i}. \tag{8.39}$$

The third numeric function maps from the relative number of customers holding the product RNp (Equation (8.9)), relative number of volumes of the product RVp (Equation (8.10)), sum of relative number of customers holding a product from the set of products of interest $SRNp$ (Equation (8.11)), and the sum of relative volumes of the products of interest $SRVp$ (Equation (8.12)), to the weight of the product w_{PR}:

$$f_{w_{PR}}(RNp, RVp, SRNp, SRVp) = 0.4 \cdot \frac{RNp}{SRNp} + 0.6 \cdot \frac{RVp}{SRVp}. \tag{8.40}$$

Model for the assessment of product/customer pair's reputational risk has two functions for transforming qualitative values to numeric (*Function type 2*) and two qualitative functions (*Function type 1*).

Function for transforming qualitative values from D_{qRI_1} into values from numeric interval $D_{nRI_1} = [1, 5]$ is modeled as:

$$f_{nRI_1}(y) = \begin{cases} 1 & \text{if } y \text{ is low} \\ 2 & \text{if } y \text{ is medium-low} \\ 3 & \text{if } y \text{ is medium} \\ 4 & \text{if } y \text{ is high} \\ 5 & \text{if } y \text{ is very high} \end{cases}. \tag{8.41}$$

The function for transforming qualitative values from D_{RV_1c} into numeric values of $D_{nRV_1c} = [1, 5]$ is:

$$f_{nRV_1c}(y) = \begin{cases} 1 & \text{if } y \text{ is low} \\ 2 & \text{if } y \text{ is medium-low} \\ 3 & \text{if } y \text{ is medium} \\ 4 & \text{if } y \text{ is high} \\ 5 & \text{if } y \text{ is very high} \end{cases}. \tag{8.42}$$

The qualitative function mapping from values of performance assessment of the product qP and the values of mismatching assessment of the customer qM, into the values of the combined values of qPM is given in the Table 8.5. The table gives an output value from D_{qPM} (the right-most column) for each combination of input values from D_{qP} and D_{qM} (first two columns).

The function f_{qRI_1} mapping from all combinations of D_{qS}, D_{qPM} and D_{qRV_1c} to D_{qRI_1} is not presented due to space constraints but it is available in (Project FIRST, 2013).

8.2.1 Relational Aggregation Functions

Relational aggregation functions in the use-case of bank reputational risk assessment denote the way on how the multiple values from product/customer are aggregated to product levels, how products influence counterpart evaluations, and finally, how multiple counterparts aggregate to the bank reputational risk evaluation. In this section all relational aggregation functions are presented.

Table 8.5: The aggregation function for combined performance and mismatching from the qualitative values of performance assessment and mismatching assessment.

qP	qM	qPM
very high	very high	very high
very high	high	very high
very high	medium	high
very high	low	medium
very high	in-line	low
high	very high	very high
high	high	high
high	medium	high
high	low	medium
high	in-line	low
medium	very high	very high
medium	high	high
medium	medium	medium
medium	low	low
medium	in-line	low
low	very high	high
low	high	high
low	medium	medium
low	low	low
low	in-line	in-line
in-line	very high	high
in-line	high	medium
in-line	medium	low
in-line	low	low
in-line	in-line	in-line

The top-most relational aggregation function $f_{RI_{BA}}$ is defined in the model BA, for attribute RI_{BA}. The relational aggregation function computes a weighted sum of counterparts' reputational risk RI_{CP} and their respective weights w_{CP}. The aggregation function's description is $f_{RI_B} : (D_{RI_{CP}})^n \times (D_{w_{CP}})^n \mapsto D_{RI_{BA}}$, and is implemented as:

$$f_{RI_{BA}}(\overrightarrow{RI_{CP}}, \overrightarrow{w_{CP}}) = \sum_{i=1}^{n} RI_{CP,i} \cdot w_{CP,i}, \tag{8.43}$$

here, $\overrightarrow{RI_{CP}} = (RI_{CP,1}, RI_{CP,2}, \ldots, RI_{CP,n})$ are single evaluations of counterpart's reputational risk and $\overrightarrow{w_{CP}} = (w_{CP,1}, w_{CP,2}, \ldots, w_{CP,n})$ are evaluations of respective counterpart's weights, where each $RI_{CP,i} \in D_{RI_{CP}}$ and each $w_{CP,i} \in D_{w_{CP}}$.

Model CP has two relational aggregation functions. The first function $f_{weighted\_RI_{PR}}$ computes a weighted sum of corresponding products' reputational risk RI_{PR} and their respective weights w_{PR}. Function's description is $f_{weighted\_RI_{PR}} : (D_{RI_{PR}})^n \times (D_{w_{PR}})^n \mapsto D_{weighted\_RI_{PR}}$, and is implemented as:

$$f_{weighted\_RI_{PR}}(\overrightarrow{RI_{PR}}, \overrightarrow{w_{PR}}) = \sum_{i=1}^{n} RI_{PR,i} \cdot w_{PR,i}, \tag{8.44}$$

here, $\overrightarrow{RI_{PR}} = (RI_{PR,1}, RI_{PR,2}, \ldots, RI_{PR,n})$ are evaluations of products' reputational risk and $\overrightarrow{w_{PR}} = (w_{PR,1}, w_{PR,2}, \ldots, w_{PR,n})$ evaluations of respective product's weights, where each $RI_{PR,i} \in D_{RI_{PR}}$ and each $w_{PR,i} \in D_{w_{PR}}$.

The second relational aggregation function $f_{w_{CP}}$ in the model CP is defined for relational attribute w_{CP}. The relational aggregation function maps from the corresponding product weights w_{PR}, to the weight of the counterpart w_{CP}, and computes the sum of the weights: $f_{w_{CP}} : (D_{w_{PR}})^n \mapsto D_{w_{CP}}$. The function is implemented as:

$$f_{w_{CP}}((w_{PR,1}, w_{PR,2}, \ldots, w_{PR,n})) = \sum_{i=1}^{n} w_{PR,i}. \tag{8.45}$$

Finally, the last model that has subordinate relational models is the model PR. The model has three relational aggregation functions. The first aggregation function f_{RI_w} aggregates numeric reputation risk of product/customer pair nRI_1 and numeric relative customer's volume of the product nRV_1c, to a sum of pairwise products, RI_w. Function's description is $f_{RI_w} : (D_{nRI_1})^n \times (D_{nRV_1c})^n \mapsto D_{RI_w}$. It is implemented as:

$$f_{RI_w}(\overrightarrow{nRI_1}, \overrightarrow{nRV_1c}) = \sum_{i=1}^{n} nRI_{1,i} \cdot nRV_1c_i, \tag{8.46}$$

where, $\overrightarrow{nRI_1} = (nRI_{1,1}, nRI_{1,2}, \ldots, nRI_{1,n})$ are single evaluations of product/customer reputational risks and $\overrightarrow{nRV_1c} = (nRV_1c_1, nRV_1c_2, \ldots, nRV_1c_n)$ are evaluations of the corresponding relative customer's volumes of the product, where each $nRI_{1,i} \in D_{nRI_1}$ and each $nRV_1c_i \in D_{nRV_1c}$.

The second relational aggregation function $f_{sum\_nV_iP_i}$ in the model for product's reputational risk assessment computes the sum $sum\_nV_iP_i$ of relative customer's volume of the product nRV_1c. The description of the function is: $f_{sum\_nV_iP_i} : (D_{nRV_1c})^n \mapsto D_{sum\_nV_iP_i}$. The function is implemented as:

$$f_{sum\_nV_iP_i}((nRV_1c_1, nRV_1c_2, \ldots, nRV_1c_n)) = \sum_{i=1}^{n} nRV_1c_i. \tag{8.47}$$

Finally, the third relational aggregation function f_{RI_u} in model PR computes an average of product/customer pair's reputational risk (RI_u). The description of the function is $f_{RI_u} : (D_{nRI_1})^n \mapsto D_{RI_u}$, and is implemented as:

$$f_{RI_u}((nRI_{1,1}, nRI_{1,2}, \ldots, nRI_{1,n})) = \frac{1}{n} \sum_{i=1}^{n} nRI_{1,i}. \tag{8.48}$$

8.3 Evaluation of Alternatives

Evaluation of alternatives was done on one of the test sets within project FIRST (Project FIRST, 2013). The alternatives are derived from a bank, *Banca Monte dei Paschi di Siena* (MPS) in the period of April 13 to May 24, 2013. The bank involves 11 counterparts, 985 products and 130565 customers, creating 327826 different product/customer pairs.

Evaluation for one day of the period, the main alternative (Banca Monte dei Paschi di Siena) was entered, adding all 11 counterparts as its relational alternatives. Additionally, each of the 985 products were assigned to one of the 11 counterparts, as a new relational alternative. Here, the values for RNp, RVp, $SRNp$ and $SRVp$ were precomputed, and entered in the corresponding product. Finally, for each of the 327826 product/customer pairs a relational alternative is created, and assigned to the corresponding product. Furthermore, the values for each product/customer for attributes qS, qP, qM and qRV_1c are precomputed.

Note here that all input values were precomputed, before entering them into the values for alternatives. However, with the custom aggregation functions, we could devise a system that would query the wanted bank and sentiment data in the form of an SQL query or issue a call to a web-service. For instance, in project FIRST (Project FIRST, 2013),

the sentiment data was available through exposed web-service, and the querying the web-service would be easily integrateable into the model. However the main bottleneck of the evaluation would become the query to the web-service, not the evaluation. For that purpose, the authors Bohanec et al. (2013, 2014) retrieved the wanted data beforehand, and precomputed the needed values.

8.4 Results

The developed model was evaluated in project FIRST (Bohanec et al., 2014; Project FIRST, 2013). Here, we provide the condensed results of the evaluation.

Evaluation was in two stages. Both evaluations were based only on opinions of expert, since no decision support system or "golden standard" was available to compare with. The first evaluation was done during development of the model, where the model and results on test data were monitored by experts from MPS. Based on experts' opinions, model was adapted several times — mostly manipulation of weights and discretization rules of the basic data processing module, and decision rules aggregation functions.

Once the model converged to a satisfactory state, the second evaluation stage was carried out. It involved ten experts who previously did not collaborate in the development of the model, of which eight experts were from MPS, one from a financial institution, and one from a consulting company. All of them had experience in risk management, auditing, communications, compliance, and customer relationship management. The experts were selected on the basis of their roles in financial institutions and their knowledge and expertise in reputational risk issues, and are considered as highly representative of potential end-users of RIM and related decision support software.

The evaluation started with presentation of RIM model to all involved experts. Descriptions of all the prototype variables and features, as well as an explanation of the purpose and details of the used questionnaire was given. Additional information was subsequently provided to enable experts to explore and further assess the prototype features.

At a later stage, each expert individually filled a questionnaire. In relation to RIM, the questionnaire addressed the representativeness of the model and, consequently, of the synthetic reputational risk indicator. Specific questions asked whether RIM was properly constructed, how effective was it in representing a bank's reputational risk exposure from selling third-party products, on how input variables were selected and how representative they were, on the type of decision rules/aggregation of variables, and on the model's representation (Bohanec et al., 2014).

Respondents expressed a positive opinion on the selection of variables, weights, decision rules and on the construction of RIM. 90% of respondents gave a positive feedback. A 90% positive opinion was also expressed on the effectiveness of the synthetic reputational risk indicator and the use of sentiment as a support for end users in their daily activities. 60% of experts believed that the availability of time series for the sentiment was extremely important, not only for the selection of the counterpart and type of product, but also for the pricing of products. This is in fact an upside potential of the model. 70% positive responses were received regarding model structure and representation. However, the respondents thought the user experience of the the prototype should be further improved — for example, the number of used information sources could be increased for better reliability.

According to the experts, the evaluation results over the used time period were in line with the expectations, however appeared to be significantly sensitive to sentiment data. In 87% of the cases, RIM assessments were consistent with the estimates provided by the end-user experts. This is a positive outcome, very much in line with the expectations for an effective tool.

8.5 Concluding Remarks

The purpose of the use-case presentation was to reimplement the model from the original study, natively in DEXx framework. The newly implemented model uses relational models and relational aggregation functions, numeric attributes and primitive function types, presented in this thesis.

Considering the criteria described in the methodological part of the introduction, we can assess the implementation of the use-case in DEXx:

- The previous implementation of the model was developed specifically for the purpose of this model only, in Java programming language — a very small part of the model (6 attributes) was developed in DEXi (Bohanec, 2015b) and ported to Java using jDEXi (Bohanec, 2015c). Implementation of the RIM model is however natively possible using DEXx framework and the presented extensions.
- Numeric attributes and relational models are a crucial part of this particular model, so these two DEXx extensions are necessary for the implementation.
- Since the original implementation of the model is programmed in Java, the running times of the evaluation are comparable, with the implementation of model using DEXx.

The important characteristics of the model for bank reputational risk assessment are the extensive use of numeric and relational attributes. This fact probably arises as a property of financial domain. Without the extensions for numeric attributes and relational attributes, such models would have been impossible to implement with DEX. Furthermore, we are not aware of any MCDM method that could allow such an easy and methodologically clean concurrent usage of qualitative and numeric variables with relationally structured inputs.

The original model developed in the course of project FIRST (Project FIRST, 2013) was implemented in the Banca Monte dei Paschi di Siena to analyse bank's reputational risk, based on the reputational risk of their customers, offered products and counterparts.

Chapter 9

Model for Sustainability Assessment of Electric Energy Production Technologies

The model for sustainability assessment of energy options is tackling a complex problem, which requires strategic planning and management for years in advance. The model was used on Slovenian specific electric energy production technologies, however it is usable for any other country with minor modifications. The study was done as a project funded by a Slovenian electrical company (B. Kontić et al., 2014b, 2016).

The main purpose of the study was to develop a model that makes a transparent and reproducible identification of reliable, rational, and environmentally sound electric energy production technologies in Slovenia by 2050 (B. Kontić, Bohanec, Trdin, et al., 2014; B. Kontić et al., 2014b; Bohanec, Trdin, & Kontić, 2015a, 2015b; B. Kontić et al., 2016). Firstly, the study considered eight electric energy production technologies for assessment. Then, the models were extended to facilitate evaluation of the so-called *technology mixtures* — here a technology mixture is some state (portfolio) in the country, giving the technologies involved and their respective shares. Finally, the assessment of technology mixtures was used in a simulation setting, where different outcomes were evaluated through time, with respect to some predefined events.

The purpose of this chapter is to present the use-case: the development of models, simulation framework, and evaluation of alternatives, using the DEXx implementation. The use-case uses the extensions of native hierarchies, relational aggregation functions and numeric attributes.

The assessment of electric energy production technologies was done in three steps, where each subsequent step included the results and the model of the previous steps:

- The first step was the development and usage of a model *Technology (T)* which was used for evaluating eight individual technologies — both conventional and renewable energy sources. The model considers three main dimensions of assessment: *Rationality*, *Feasibility* and *Uncertainties* involved in the technology.
- The second step was to incorporate the evaluations of individual technologies into the evaluation of *Technology mixture (M)* — collection of technologies, considering relative shares of each technology in the total installed capacity. The two main dimensions of technology mixture assessment are *Reasonability* and *Long-term appropriateness*.
- The third step was to use the model for evaluation of technology mixtures and use

the model in a simulation setting. The *Simulation model (S)* was run from years 2013 to year 2050, considering 8 different potential management events, which resulted in 64 different scenarios. An event in a scenario is considered a shutting-down of an existing power plant or construction of a new one.

The first step was achieved by developing a purely qualitative multi-criteria DEX model, creating eight alternatives (technologies). The input values were assigned to alternatives based on empirical research (B. Kontić et al., 2014b, 2016) of energy production options in Slovenia.

The second step was achieved with an additional multi-criteria model M, which was initially developed in DEXi (Bohanec, 2015b), but was then transferred into DEXx implementation, due to the need of numeric attributes. Additionally, the model for single technology assessment T is used as a relational model, to model M. Model M has eight relational attributes, whose counterpart input attributes are all from model T.

Models for technology mixture and single technology assessment are relationally structured as presented in Figure 9.1. The figure also indicates the simulation model, using the technology mixture and single technology assessment models. Multiple evaluations of technologies influence a single technology mixture assessment (through relational attributes rx and counterpart relational inputs ox).

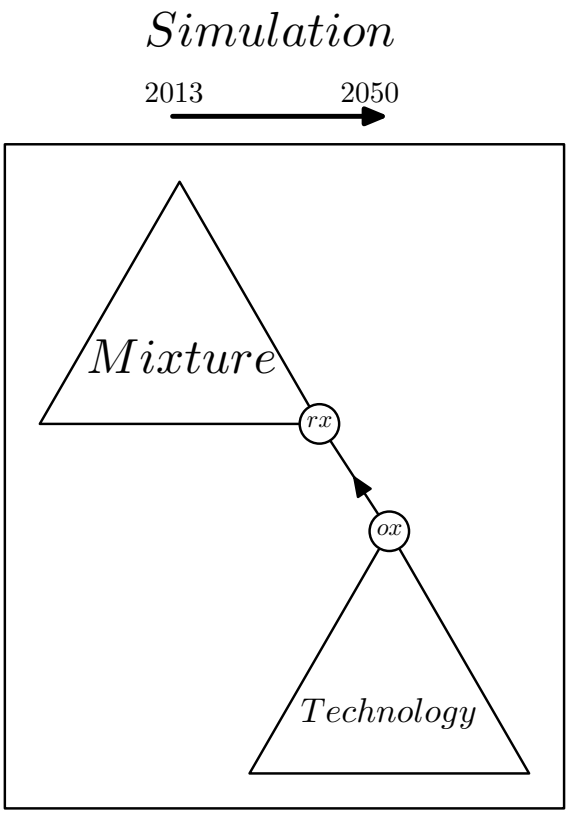

Figure 9.1: The configuration of main model for technology mixture assessment (*Mixture*) with the relationally connected model for single technology assessment (*Technology*) assessment. Each technology mixture assessment depends on multiple technology assessments. The *Simulation* model wraps both models and runs the simulation from year 2013 to 2050.

The third step was creation of model S with a direct implementation in Java, without using any framework. The model's inputs are N independent events and respective

consequences, and some beginning state s_0 of the simulation (given as one alternative representing the technology mixture). The simulation model generates 2^N different scenarios, and evaluates s_0 through years 2013–2050, considering the consequences of the selected scenario.

Model T has eight fixed alternatives, conventional and renewable sources of energy: coal fired, gas fired, biomass fired, oil fired, nuclear, hydro, wind, and photovoltaic. When evaluating them in the context of model T, we are evaluating each technology individually, without considering their placement in an actual mixture of technologies.

Model M has seven alternatives defined, where one of the alternatives presents the technology mixture that was used in Slovenia in 2013. Other six alternatives present some hypothetical scenarios which are interesting to the expert. The relational alternatives are the energy production technologies, corresponding to the scenario. The input attributes for relational alternatives are the same as in the evaluation with model T. However, we also introduce an additional input attribute, which represents the actual respective share in the technology mixture (according to the relative installed capacity). The weighting attribute is used in the relational aggregation of values.

9.1 Model Description and Structure

The model for technology assessment T (presented in Table 9.1) was initially implemented in DEXi, and after that ported to the DEXx implementation for usage in relational aggregation. The model was using DEXi "links" for modeling two hierarchical connections — during the initial import to DEXx methodology, the model was created as a full hierarchy, omitting the "links". The model considers three main dimensions of assessment: *Rationality*, *Feasibility* and *Uncertainties* involved in the technology. *Rationality* is further split into assessment of *Contribution to development*, *Economy* and *Land use and pollution*. *Feasibility* of a technology is considered in three aspects: *Technical feasibility*, *Economic feasibility* and *Spatial feasibility*. *Technological dependence*, *Possible changes* and *Perception of risks* all influence the *Uncertainties* involved in the assessment of technology. These lower branches are further split into more detailed assessments of each concept. The model has 65 attributes in total — 37 of those are aggregated attributes and 28 are input attributes. The model also contains two attributes that are hierarchically connected.

On the left side of the table, the model structure is presented. The top-most attribute is the root of the model. Note that the model has two roots: *Technology* and *Weight*, however for the technology assessment only *Technology* root is interesting. *Weight* is used during evaluation of technology mixtures and denotes the importance of the respective evaluations of alternatives in model T. Each attribute's direct descendants are connected with a line and indented to the right. Attributes displayed in green text are the hierarchical attributes — attributes which influence multiple parts of the same model and subsequently have multiple parent attributes (see Section 4.1). These attributes are *Contribution to development* and *Licenses*.

On the right side of the table, the scales for corresponding attributes are given. All scales are preferentially ordered in an increasing order. For the qualitative values, the worst values are written in red, the best values are written in green, and the intermediate values are written in black text. The numeric attribute *Weight* however has a numeric scale, which is also ordered increasingly (higher numeric value denotes greater importance in relational aggregation).

Formally, the model for technology assessment T has structural properties derived from Table 9.1. The attributes X are:

Table 9.1: The model for assessing a single technology. On the left side, the structure of the attributes is presented, and on the right side, their corresponding scales are displayed. The scales are ordered from worst values (displayed in red) to best values (displayed in green).

Attribute	Scale
Technology	unsuitable, weak, suitable, good, excellent
Rationality	inappropriate, low, medium, high
Contribution to development	low, medium, high
Economic	low, medium, high
Societal	low, medium, high
Economic-Technical advancement	low, medium, high
Technical level	low, medium, high
Expected development	low, medium, high
Economy	low, medium, high
Financial aspects	less suitable, suitable, more suitable
Energy price	high, medium, low
Financing	less suitable, suitable, more suitable
Financial sources	uncertain, less certain, certain
Financial shares	less suitable, suitable, more suitable
Long-term liabilities	less suitable, suitable, more suitable
Efficiency	low, medium, high
Energy ratio	low, medium, high
Return period	long, medium, short
Independence	low, medium, high
Dependence	very high, high, medium, low, none
Land use and pollution	unsuitable, less suitable, suitable, more suitable
Spatial availability	less suitable, suitable, more suitable
Land availability	low, medium, high
Energy share provision	low, medium, high
Resource protection	weak, present, effective
Water protection	weak, present, effective
Land protection	weak, present, effective
Landscape protection	weak, present, effective
Pollution	high, medium, low
Health impact	high, medium, low
Air pollution	high, medium, low
Greenhouse gases	high, medium, low
Other pollutants	high, medium, low
Public health status	low, medium, high
Contribution to development	low, medium, high
Feasibility	low, medium, high
Technical feasibility	low, medium, high
Technological complexity	less suitable, suitable, more suitable
Infrastructure availability	low, medium, high
Accessibility	low, medium, high
Fuel availability	low, medium, high
Fuel accessibility	low, medium, high
Economic feasibility	low, medium, high
Investment feasibility	low, medium, high
Return of investment	less suitable, suitable, more suitable
Spatial feasibility	low, medium, high
Societal feasibility	low, medium, high
Social acceptance	low, medium, high
Permitting	no, yes
Spatial suitability	low, medium, high
Uncertainties	very high, high, medium, low, none
Technological dependence	very high, high, medium, low, none
Foreign dependence	very high, high, medium, low, none
Construction	high, medium, low
Licences	strong restrictions, moderate restrictions, no restrictions
Contracts	strong restrictions, moderate restrictions, no restrictions
Special materials	strong restrictions, moderate restrictions, no restrictions
Operation	high, medium, low
Licences	strong restrictions, moderate restrictions, no restrictions
Weather dependence	high, medium, low
Fuel supply dependence	high, medium, low
Political stability	no, low, high
Possible changes	negative, no, positive
Possible societal changes	negative, no, positive
Possible world changes	negative, no, positive
Perception of risks	very high, high, medium, low, none
Weight	$[0, 1]$

$$X = \{\text{Technology}, \text{Rationality}, \text{Contribution to development}, \ldots, \text{Weight}\}. \tag{9.1}$$

The descendants of attributes are described by mapping S, and are given as follows:

$$\begin{gathered} S(\text{Technology}) = \{\text{Rationality}, \text{Feasibility}, \text{Uncertainties}\}, \\ S(\text{Rationality}) = \{\text{Contribution to development}, \text{Economy}, \\ \text{Land use and pollution}\}, \\ S(\text{Contribution to development}) = \{\text{Economic}, \text{Societal}, \\ \text{Economic-Technical advancement}\}, \\ \ldots, \\ S(\text{Public health status}) = \{\text{Contribution to development}\}, \\ \ldots, \\ S(\text{Economic}) = \{\}, \\ S(\text{Societal}) = \{\}, \\ \ldots, \\ S(\text{Weight}) = \{\}. \end{gathered} \tag{9.2}$$

Now, the descendants S are defined, also the $aggAttributes_{\text{T}}$ can be determined as:

$$\begin{gathered} aggAttributes_{\text{T}} = \{\text{Technology}, \text{Rationality}, \text{Contribution to development}, \ldots, \\ \text{Possible changes}\}. \end{gathered} \tag{9.3}$$

Furthermore, $modelInputs_{\text{T}}$ can be defined:

$$modelInputs_{\text{T}} = \{\text{Economic}, \text{Societal}, \text{Technical level}, \ldots, \text{Weight}\}. \tag{9.4}$$

As far as the structure of the model goes, only the $modelOutputs_{\text{T}}$ are determined as:

$$modelOutputs_{\text{T}} = \{\text{Technology}, \text{Weight}\}. \tag{9.5}$$

The scales of the attributes are also presented on the right side of the Table 9.1. The scales of the attributes are defined by D_x, where $x \in X$:

$$\begin{gathered} D_{\text{Technology}} = (\text{unsuitable}, \text{weak}, \text{suitable}, \text{good}, \text{excellent}), \\ D_{\text{Rationality}} = (\text{inappropriate}, \text{low}, \text{medium}, \text{high}), \\ D_{\text{Contribution to development}} = (\text{low}, \text{medium}, \text{high}), \\ \ldots, \\ D_{\text{Perception of risks}} = (\text{very high}, \text{high}, \text{medium}, \text{low}, \text{none}), \\ D_{\text{Weight}} = [0, 1]. \end{gathered} \tag{9.6}$$

The model structure and qualitative aggregation functions for technology mixture assessment M (presented in Table 9.2) were implemented using DEXi, and after that ported to the DEXx methodology, in order to introduce relational aggregation functions and the relational model T. The two main dimensions of technology mixture assessment are *Reasonability* and *Long-term appropriateness*. *Reasonability* is further split into *Energy demand coverage* and *Feasibility and rationality*. *Fulfilment of goals and interests* and *Lifetime of supply* both influence the *Long-term appropriateness*. These attributes are similarly split into more basic concepts, which can be assessed on more refined attributes of the

technology mixture. The model has 27 attributes in total — 12 of those are aggregated attributes and 15 are input attributes. Out of 15 input attributes, there are 8 relational input attributes and 7 basic input attributes.

On the left side of the table, the model structure is presented. There is only one root attribute: *Technology mixture*. Attributes written in blue text are the relational attributes (*Health impacts*, *Possible changes*, *Feasibility*, *Economy*, *Low carbon*, *Rational land use*, *Nature protection* and *Independence*). The descendants of relational attributes are attributes from subordinate relational model T. The respective descendants are *Health impact*, *Possible changes*, *Feasibility*, *Economy*, *Greenhouse gases*, *Spatial availability*, *Resource protection* and *Independence*. Each relational attribute also has the *Weight* as its counterpart relational input.

On the right side, the scales for the attributes are presented. All scales are preferentially ordered in an increasing order. For the qualitative values, the worst values are written in red, the best values are written in green, and the intermediate values are written in black text.

Similarly, attributes X, successors S, aggregated attributes $aggAttributes_{\mathrm{M}}$, model inputs $modelInputs_{\mathrm{M}}$, model outputs $modelOutputs_{\mathrm{M}}$ and scales D_x, are constructed from Table 9.2 for the model for technology mixture assessment.

$$X = \{\text{Technology mixture}, \text{Reasonability}, \text{Energy demand coverage}, \ldots, \text{Lifetime of supply}\}, \tag{9.7}$$

$$\begin{gathered} S(\text{Technology mixture}) = \{\text{Reasonability}, \text{Long-term appropriateness}\}, \\ S(\text{Reasonability}) = \{\text{Energy demand coverage}, \text{Feasibility and rationality}\}, \\ S(\text{Energy demand coverage}) = \{\text{Reliability of supply}, \text{Uncertainties}\}, \\ \ldots, \\ S(\text{Installed capacity}) = \{\}, \\ S(\text{Energy produced}) = \{\}, \\ \ldots, \\ S(\text{Lifetime of supply}) = \{\}, \end{gathered} \tag{9.8}$$

$$aggAttributes_{\mathrm{M}} = \{\text{Technology mixture}, \text{Reasonability}, \text{Energy demand coverage}, \ldots, \text{Energy users capabilities}\}, \tag{9.9}$$

$$modelInputs_{\mathrm{M}} = \{\text{Installed capacity}, \text{Energy produced}, \text{Base load}, \ldots, \text{Lifetime of supply}\}, \tag{9.10}$$

$$modelOutputs_{\mathrm{M}} = \{\text{Technology mixture}\}, \tag{9.11}$$

$$\begin{gathered} D_{\text{Technology mixture}} = (\text{unsuitable}, \text{weak}, \text{suitable}, \text{good}, \text{excellent}), \\ D_{\text{Reasonability}} = (\text{unreasonable}, \text{less-reasonable}, \text{reasonable}, \text{desired}), \\ D_{\text{Energy demand coverage}} = (\text{low}, \text{medium}, \text{good}, \text{high}), \\ \ldots, \\ D_{\text{Lifetime of supply}} = (\text{short}, \text{medium}, \text{long}). \end{gathered} \tag{9.12}$$

The model for technology mixture assessment has a subordinate relational model (model for technology assessment T), and hence has eight relational attributes (rx):

Table 9.2: The model for assessing technology mixtures. On the left side, the structure of the attributes is presented, and on the right side, their corresponding scales are displayed. The scales are ordered from worst values (displayed in red) to best values (displayed in green). Furthermore, the *Weight* attribute has a numeric interval scale, which is also preferentially ordered.

Attribute	Scale
Technology mixture	unsuitable, weak, suitable, good, excellent
Reasonability	unreasonable, less-reasonable, reasonable, desired
Energy demand coverage	low, medium, good, high
Reliability of supply	low, medium, high, very high
Availability	low, medium, high
Installed capacity	unsuitable, suitable, exceed
Energy produced	unsuitable, suitable, exceed
Base load	low, medium, high
Peaks	no, yes
Uncertainties	very high, high, medium, low
Health impacts	high, medium, low
Health impact	high, medium, low
Weight	$[0, 1]$
Possible changes	negative, no, positive
Possible changes	neg, no, pos
Weight	$[0, 1]$
Feasibility and rationality	weak, low, medium, high
Feasibility	low, medium, high
Feasibility	low, medium, high
Weight	$[0, 1]$
Economy	low, medium, high
Economy	low, medium, high
Weight	$[0, 1]$
Long-term appropriateness	low, medium, high
Fulfilment of goals and interests	low, medium, high
Environmental goals	low, medium, high
Low carbon	low, medium, high
Greenhouse gases	high, medium, low
Weight	$[0, 1]$
Rational land use	low, medium, high
Spatial availability	less suitable, suitable, more suitable
Weight	$[0, 1]$
Nature protection	low, medium, high
Resource protection	weak, present, effective
Weight	$[0, 1]$
National interests	low, medium, high
Independence	low, medium, high
Independence	low, medium, high
Weight	$[0, 1]$
Energy users capabilities	low, medium, high
Energy supply to all	low, medium, high
Protection of vulnerable groups	low, medium, high
Lifetime of supply	short, medium, long

1. Attribute *Health impacts* has two counterpart relational inputs: assessment of technology's health impact (*Health impact*) and the respective weight of the technology (*Weight*).

2. *Possible changes* has two counterpart relational inputs: possible changes the technology might introduce (*Possible changes*) and the technology's (*Weight*).

3. *Feasibility* of technology mixture is assessed through assessment's of individual technology's *Feasibility* and respective technology's (*Weight*).

4. Similarly, the *Economy* is assessed through assessment's of individual technology's economic aspects (*Economy*) and its (*Weight*).

5. *Low carbon* relational attribute is assessed from individual technology's contribution to *Greenhouse gases* and the *Weight*.

6. *Rational land use* is assessed from *Spatial availability* of technologies and the *Weight* of respective technology.

7. *Nature protection* gives the overall value of *Resource protection* for individual technologies, and is also influenced by the *Weight* of the respective technology.

8. Relational attribute *Independence* computes the degree of independence of the technology mixture, from the individual technology's *Independence* and the respective *Weight* of the technology.

Here note that none of the relational attributes actually uses the overall final evaluation of model T (attribute *Technology*). Rather they use more specific evaluations from the lower parts of the model T. With that, the relational inputs to model M aggregate respective values at higher resolution and subsequently provide more detailed evaluations of technology concepts.

Furthermore, generally all relational attributes, which are all influenced by the technologies *Weight*'s, produce a probabilistic distribution over qualitative values of respective relational attributes. This inherently means that the majority of evaluations in model M will be evaluated probabilistically and in turn produce a qualitative probabilistic distribution. This in general means that the evaluation of technology mixture will be a distribution over values of $D_{\text{Technology mixture}}$.

9.2 Aggregation Functions

This section presents some selected aggregation functions used in the sustainability assessment of energy options. Models T and M have only functions of *type 1* and relational aggregation functions, which are computing the distributions of values. Relational aggregation functions are presented in Section 9.2.1. Due to large space requirements for presentation of qualitative aggregation functions, we present only three functions: $f_{\text{Technology}}$, $f_{\text{Rationality}}$ and $f_{\text{Technology mixture}}$.

Function $f_{\text{Technology}}$ is a *function type 1*, which assigns the final qualitative value to the *Technology* attribute, based on the values of attributes *Rationality*, *Feasibility* and *Uncertainties*. The function is used in the root attribute of model T. Table 9.3 shows the rules used in the function.

The aggregation function $f_{\text{Rationality}}$ is a *function type 1*, which considers values of *Contribution to development*, *Economy*, and *Land use and pollution* to assign a value from the domain of *Rationality*. The aggregation function is used in an attribute below the root attribute of model T. Table 9.4 shows all input value combinations with respective outputs of the function.

The third function $f_{\text{Technology mixture}}$ is again a *function type 1*, which appears at the root attribute of model M. It considers two aspects of a particular technology mixture — *Reasonability* and *Long-term appropriateness*. The aggregation function is shown in Table 9.5.

9.2.1 Relational Aggregation Functions

The model for sustainability assessment of energy options has eight relational aggregation functions. All such functions are used in model M to provide inputs. The respective relational attributes are: *Health impacts*, *Possible changes*, *Feasibility*, *Economy*, *Low carbon*, *Rational land use*, *Nature protection* and *Independence*. The corresponding counterpart relational inputs from model T are *Health impact*, *Possible changes*, *Feasibility*, *Economy*,

Table 9.3: The aggregation function for technology assessment defines a mapping from every combination of input attribute values for *Rationality*, *Feasibility* and *Uncertainties*, into values from scale of *Technology*.

Rationality	*Feasibility*	*Uncertainties*	*Technology*
inappropriate	low	very high	unsuitable
inappropriate	low	high	unsuitable
inappropriate	low	medium	unsuitable
inappropriate	low	low	unsuitable
inappropriate	low	none	unsuitable
inappropriate	medium	very high	unsuitable
inappropriate	medium	high	unsuitable
inappropriate	medium	medium	unsuitable
inappropriate	medium	low	unsuitable
inappropriate	medium	none	unsuitable
inappropriate	high	very high	unsuitable
inappropriate	high	high	unsuitable
inappropriate	high	medium	unsuitable
inappropriate	high	low	unsuitable
inappropriate	high	none	unsuitable
low	low	very high	unsuitable
low	low	high	weak
low	low	medium	weak
low	low	low	suitable
low	low	none	suitable
low	medium	very high	unsuitable
low	medium	high	suitable
low	medium	medium	good
low	medium	low	good
low	medium	none	good
low	high	very high	weak
low	high	high	suitable
low	high	medium	good
low	high	low	good
low	high	none	good
medium	low	very high	unsuitable
medium	low	high	weak
medium	low	medium	weak
medium	low	low	suitable
medium	low	none	suitable
medium	medium	very high	weak
medium	medium	high	suitable
medium	medium	medium	good
medium	medium	low	good
medium	medium	none	excellent
medium	high	very high	weak
medium	high	high	suitable
medium	high	medium	good
medium	high	low	excellent
medium	high	none	excellent
high	low	very high	weak
high	low	high	weak
high	low	medium	weak
high	low	low	suitable
high	low	none	good
high	medium	very high	weak
high	medium	high	suitable
high	medium	medium	good
high	medium	low	excellent
high	medium	none	excellent
high	high	very high	weak
high	high	high	suitable
high	high	medium	good
high	high	low	excellent
high	high	none	excellent

Table 9.4: The aggregation function for rationality assessment defines a mapping from every combination of input attribute values for *Contribution to development*, *Economy*, and *Land use and pollution*, into values from scale of *Rationality*.

Contribution to development	*Economy*	*Land use and pollution*	*Rationality*
low	low	unsuitable	inappropriate
low	low	less suitable	inappropriate
low	low	suitable	inappropriate
low	low	more suitable	inappropriate
low	medium	unsuitable	inappropriate
low	medium	less suitable	inappropriate
low	medium	suitable	medium
low	medium	more suitable	medium
low	high	unsuitable	inappropriate
low	high	less suitable	inappropriate
low	high	suitable	medium
low	high	more suitable	high
medium	low	unsuitable	inappropriate
medium	low	less suitable	inappropriate
medium	low	suitable	low
medium	low	more suitable	low
medium	medium	unsuitable	inappropriate
medium	medium	less suitable	low
medium	medium	suitable	medium
medium	medium	more suitable	high
medium	high	unsuitable	inappropriate
medium	high	less suitable	low
medium	high	suitable	high
medium	high	more suitable	high
high	low	unsuitable	inappropriate
high	low	less suitable	low
high	low	suitable	medium
high	low	more suitable	medium
high	medium	unsuitable	inappropriate
high	medium	less suitable	low
high	medium	suitable	high
high	medium	more suitable	high
high	high	unsuitable	inappropriate
high	high	less suitable	low
high	high	suitable	high
high	high	more suitable	high

Table 9.5: The aggregation function for technology mixture assessment defines an output value for every combination of input attribute values for *Reasonability* and *Long-term appropriateness*, into values from scale of *Technology mixture*.

Reasonability	*Long-term appropriateness*	*Technology mixture*
unreasonable	low	unsuitable
unreasonable	medium	unsuitable
unreasonable	high	unsuitable
less reasonable	low	unsuitable
less reasonable	medium	suitable
less reasonable	high	suitable
reasonable	low	unsuitable
reasonable	medium	good
reasonable	high	excellent
desired	low	weak
desired	medium	excellent
desired	high	excellent

Greenhouse gases, *Spatial availability*, *Resource protection* and *Independence*. Furthermore, every relational attribute also has a second counterpart relational input from model

T — *Weight*.

All relational aggregation functions map from an extended domain over qualitative value scales and a numeric value (from interval $[0, 1]$) to an extended domain over qualitative value scale of the relational attribute. The prototype relational aggregation function appearing in model M is presented in Algorithm 9.1. Firstly, the respective scales are retrieved for the relational attribute X and the first counterpart attribute — we assume that the first attribute is the attribute holding the evaluation (line 2). Then, for each evaluation v_i (generally a probabilistic value distribution), the algorithm iterates over all $(value, probability)$ pairs present in v_i. Lines 7–8 find the aggregated value *aggValue* (in the domain of X), to which the evaluated *value* is mapped, respective to the positional index. Here note that this step is necessary because relational attributes *Possible changes*, *Low carbon*, *Rational land use* and *Nature protection*, have different scales than their respective counterpart input attributes. Finally, the *aggValue*'s probability is increased by the original *probability* multiplied by the respective weight w_i of the alternative. There is no need to normalize the final value, since probabilities in line 6 sum to 1, and sum of w_is is also equal to 1 (see line 20 in Algorithm 9.3). Hence all calculations can be interpreted probabilistically.

Algorithm 9.1: Algorithm represents the prototype relational aggregation function f_X used in model M, for attribute X. The function weighs each appearing evaluation with the respective weight of the alternative. Note that evaluations of relational alternatives and the final evaluation are generally probabilistic value distributions (not crisp qualitative values) and need to be handled accordingly.

Input: Evaluations of relational alternatives $V = (v_1, v_2, \ldots, v_n)$,
Weights of relational alternatives $W = (w_1, w_2, \ldots, w_n)$
Output: Final evaluation of relational alternative

```
1  scale = X.scale();
2  counterpartAttribute = X.counterpartAttributes()[0];
3  counterpartScale = counterpartAttribute.scale();
4  EVALUATION = X.emptyDistributionValue();
5  for i = 1; i ≤ |V|; i = i + 1 do                    /* For each evaluation */
6  |  for (value, probability) ∈ V[i] do               /* For value in evaluation */
7  |  |  index = counterpartScale.index(value);
8  |  |  aggValue = scale[index];
9  |  |  EVALUATION.increaseProbability(aggValue, probability · w_i);
10 |  end
11 end
12 return simplify(EVALUATION);
```

The relational aggregation functions used in model M aggregate relational evaluations with the same prototype function presented in Algorithm 9.1. Each relational aggregation function's purpose is to weigh the relational evaluations with the respective weights of the relational alternatives (proportional shares of installed powers of technologies in a technology mixture). Since the evaluations of relational alternatives in model T are generally probabilistic value distributions, the probabilities of respective values in each evaluation also have to influence the evaluation of the relational aggregation function.

9.3 Simulation Procedure

Simulation model S is using the initial alternative s_{2013} for the simulation. Then for each of the generated scenarios (from N events), the simulation evaluates the technology mixture s_i, for $i \in [2013, 2050]$ on model M, considering the scenario's consequences. The consequences of the scenario are the joined consequences of each event, occurring in the scenario. The consequences of each event in the simulation model S are of two types:

1. Increase/decrease of the installed power of one of the eight energy production technologies;
2. Increase/decrease of the produced energy by one of the eight energy production technologies.

The simulation study considers six *events* that may occur during the course of events through the years:

- *TEŠ5* considers Unit 5 of the Thermal Plant Šoštanj. It is scheduled to be brought down in year 2023, however its life-span might be prolonged until year 2027.
- *JEK1* is concerned with Unit 1 of the Nuclear Plant Krško — it may be brought down in year 2023.
- *JEK2* is concerned with Unit 2 of Nuclear Plant Krško, which considers building the Unit 2, and running from year 2025 onwards.
- *HESM* and *HESL* events concern the building and deployment of Hydro Plants on middle and lower Sava river, respectively. The deployment might occur in years 2035 and 2025, respectively.
- *GAS* event considers the building and deployment of Gas-powered Plant in year 2025.

Each of the events modifies the total energy produced and total capacity installed, with the functions $\Delta\text{ENERGY}_X(year, event)$ and $\Delta\text{POWER}_X(year, event)$. Here the functions return the absolute value of additional energy (in GWh) and absolute value of additional power (in MW) in the given *year*, assuming the *event* will happen. The used absolute numbers are given in the Equations (9.13) through (9.24).

$$\Delta\text{ENERGY}_{\text{TEŠ5}}(year, event) = \begin{cases} -1\,656 & \text{if } year \geq 2023 \text{ and } event = \text{False} \\ -1\,656 & \text{if } year \geq 2027 \text{ and } event = \text{True} \\ 0 & \text{otherwise} \end{cases} \tag{9.13}$$

$$\Delta\text{POWER}_{\text{TEŠ5}}(year, event) = \begin{cases} -345 & \text{if } year \geq 2023 \text{ and } event = \text{False} \\ -345 & \text{if } year \geq 2027 \text{ and } event = \text{True} \\ 0 & \text{otherwise} \end{cases} \tag{9.14}$$

$$\Delta\text{ENERGY}_{\text{JEK1}}(year, event) = \begin{cases} -2\,520 & \text{if } year \geq 2023 \text{ and } event = \text{True} \\ 0 & \text{otherwise} \end{cases} \tag{9.15}$$

$$\Delta\text{POWER}_{\text{JEK1}}(year, event) = \begin{cases} -700 & \text{if } year \geq 2023 \text{ and } event = \text{True} \\ 0 & \text{otherwise} \end{cases} \tag{9.16}$$

$$\Delta\text{ENERGY}_{\text{JEK2}}(year, event) = \begin{cases} 11\,520 & \text{if } year \geq 2025 \text{ and } event = \text{True} \\ 0 & \text{otherwise} \end{cases} \tag{9.17}$$

$$\Delta\text{POWER}_{\text{JEK2}}(year, event) = \begin{cases} 1\,600 & \text{if } year \geq 2025 \text{ and } event = \text{True} \\ 0 & \text{otherwise} \end{cases} \tag{9.18}$$

$$\Delta\text{ENERGY}_{\text{HESM}}(year, event) = \begin{cases} 1\,122 & \text{if } year \geq 2035 \text{ and } event = \text{True} \\ 0 & \text{otherwise} \end{cases} \tag{9.19}$$

$$\Delta\text{POWER}_{\text{HESM}}(year, event) = \begin{cases} 330 & \text{if } year \geq 2035 \text{ and } event = \text{True} \\ 0 & \text{otherwise} \end{cases} \tag{9.20}$$

$$\Delta\text{ENERGY}_{\text{HESL}}(year, event) = \begin{cases} 252 & \text{if } year \geq 2025 \text{ and } event = \text{True} \\ 0 & \text{otherwise} \end{cases} \tag{9.21}$$

$$\Delta\text{POWER}_{\text{HESL}}(year, event) = \begin{cases} 74 & \text{if } year \geq 2025 \text{ and } event = \text{True} \\ 0 & \text{otherwise} \end{cases} \tag{9.22}$$

$$\Delta\text{ENERGY}_{\text{GAS}}(year, event) = \begin{cases} 3\,000 & \text{if } year \geq 2025 \text{ and } event = \text{True} \\ 0 & \text{otherwise} \end{cases} \tag{9.23}$$

$$\Delta\text{POWER}_{\text{GAS}}(year, event) = \begin{cases} 600 & \text{if } year \geq 2025 \text{ and } event = \text{True} \\ 0 & \text{otherwise} \end{cases} \tag{9.24}$$

Since there are six *events* that can happen or not in each *scenario*, they collectively make $2^6 = 64$ scenarios, which are considered in the simulation study.

Model S is heavily based on the amount of *energy produced* EP by the technology mixture, amount of *power installed* PI, and amount of *power installed in the band* PB, in a given year. The amount of energy produced EP in the sum over all energies produced by a single technology, modified by the scenario ($\Delta ENERGY_X$). The amount of power installed PI is the sum of installed powers over all single technologies, modified by the scenario ($\Delta POWER_X$). The amount of power installed PB in the band is the sum of installed powers by hydro, coal, gas, oil and nuclear technologies, modified by the scenario ($\Delta POWER_X$) (B. Kontić et al., 2014b; Bohanec et al., 2015a, 2015b; B. Kontić et al., 2016).

9.3.1 Simulation Inputs

To compute the input values for model T (in scope of the simulation), the *need for energy* EN, *need for installed power* NPI and *need for installed power in the band* NPB, in a given year need to be established. The required need for produced energy in year 2013 is 12700 GWh, with an estimated yearly increase of 2.2%. Further, to cover the additional needs of energy consumption, the need is increased by 1000 GWh. The need for installed power up to year 2025 is 1700 MW, and after that 2200 MW — to cover the additional needs for installed power, the limit is further increased by 100 MW. The requirements for the band are 1000 MW to year 2025 and 1200 MW after the year 2025. These constraints are given in the Equations (9.25) through (9.27), and are valid for years from 2013 onwards.

$$\begin{aligned} \mathrm{EN}_{\mathrm{base}}(year) &= 12\,700 \cdot 0.02^{year-2013} \\ \mathrm{EN}_{\mathrm{additional}}(year) &= 13\,700 \cdot 0.02^{year-2013} \end{aligned} \tag{9.25}$$

$$\begin{aligned} \mathrm{NPI}_{\mathrm{base}}(year) &= \begin{cases} 1\,700 & \text{if } year \leq 2025 \\ 2\,200 & \text{otherwise} \end{cases} \\ \mathrm{NPI}_{\mathrm{additional}}(year) &= \begin{cases} 1\,800 & \text{if } year \leq 2025 \\ 2\,300 & \text{otherwise} \end{cases} \end{aligned} \tag{9.26}$$

$$\mathrm{NPB}(year) = \begin{cases} 1\,000 & \text{if } year \leq 2025 \\ 1\,200 & \text{otherwise} \end{cases} \tag{9.27}$$

Inputs for the model T are of two types: regular inputs assigned to the alternatives and inputs computed by the relational aggregation functions. The computation of the latter are covered in Section 9.2.1. The seven input attributes that have regular assigned input values are: *Installed capacity*, *Energy produced*, *Base load*, *Peaks*, *Energy supply to all*, *Protection of vulnerable groups* and *Lifetime of supply*. Values for these attributes can be assessed manually for the purpose of single assessment of model T, however, their computation must be formalized and implemented in a programming language, for the usage in simulation model S.

Installed capacity gets its value by summation of all installed powers of all considered technologies, modified by the $\Delta\mathrm{POWER}_X$, defined by the scenario S. Suppose, currently considered are technologies T, where each technology $t \in T$ has capacity $capacity(t)$. Then given the year y of simulation and scenario S, the value of *Installed capacity* is:

$$\text{Installed capacity}(T, S, y) = \sum_{t \in T} capacity(t) + \sum_{event \in S} \Delta\mathrm{POWER}_{event}(y, event). \tag{9.28}$$

Here, $event \in S$ is $True$ if the event is present in the scenario S, and $False$ otherwise.

A qualitative value of *Installed capacity* is assigned based on the values of actual installed capacity Installed capacity(T, S, y), and values of base and additional installed capacities $\mathrm{NPI}_{\mathrm{base}}$ and $\mathrm{NPI}_{\mathrm{additional}}$:

$$\begin{aligned} &\text{QInstalled capacity}(T, S, y) = \\ &= \begin{cases} \text{unsuitable}, & \text{Installed capacity}(T, S, y) \leq \mathrm{NPI}_{\mathrm{base}}(y) \\ \text{suitable}, & \text{Installed capacity}(T, S, y) \leq \mathrm{NPI}_{\mathrm{additional}}(y) \\ \text{exceed}, & \text{otherwise} \end{cases}. \end{aligned} \tag{9.29}$$

Similarly, the value for *Energy produced* gets an assigned value with:

$$\begin{aligned} &\text{QEnergy produced}(T,S,y) = \\ &= \begin{cases} \text{unsuitable}, & \text{Energy produced}(T,S,y) \leq \text{EN}_{\text{base}}(y) \\ \text{suitable}, & \text{Energy produced}(T,S,y) \leq \text{EN}_{\text{additional}}(y) \\ \text{exceed}, & \text{otherwise} \end{cases}, \end{aligned} \tag{9.30}$$

where Energy produced(T,S,y) is computed as the sum of produced energies $energy(t)$, for each technology t in T and given the scenario S. Concretely, the value is computed as:

$$\text{Energy produced}(T,S,y) = \sum_{t \in T} energy(t) + \sum_{event \in S} \Delta\text{ENERGY}_{event}(y, event). \tag{9.31}$$

The value for *Base load* has a more complicated scheme of computation. Firstly, the value for installed capacity of band BAND(T,S,y) is computed, which presents the sum of installed capacities from $B = \{\text{hydro}, \text{coal}, \text{gas}, \text{oil}, \text{nuclear}\}$ technologies, modified by the scenario S:

$$\text{BAND}(T,S,y) = \sum_{t \in (T \cap B)} capacity(t) + \sum_{event \in (S \cap B)} \Delta\text{POWER}(y, event). \tag{9.32}$$

Here, $(S \cap B)$ are the events that influence one of the technologies in B.

Next, the value for maxLoss(T,S,y) is computed, where its value represents the capacity installed in the scenario, if the most powerful technology is turned down or not available. The value is computed as per the following Equation:

$$\begin{aligned} &\text{maxLoss}(T,S,y) = \text{BAND}(T,S,y) - \\ &- \max_{t \in (B \cap T)} (capacity(t) + \sum_{event \in (S \cap t)} \Delta\text{POWER}(y, event)). \end{aligned} \tag{9.33}$$

Here, $event \in (S \cap t)$ represents a set of events, where *event* is from scenario S and influences the technology t.

Finally, the qualitative value for *Base load* can be determined:

$$\begin{aligned} &\text{QBase load}(T,S,y) = \\ &= \begin{cases} \text{exceed}, & \text{maxLoss}(T,S,y) > \text{NPB}(y) \\ \text{suitable}, & \text{BAND}(T,S,y) > \text{NPB}(y) \\ \text{unsuitable} & \text{otherwise} \end{cases}. \end{aligned} \tag{9.34}$$

The *Peaks* attribute represents a value, if the technology mixture is able to sustain daily peaks of energy need. To establish a value for the attribute, firstly a set of technologies, which are able to sustain peaks is determined. These technologies are: hydro, coal, gas, oil and nuclear. Let the $PEAKS$ be the set of these technologies. The set of technologies, able to sustain peaks PEAKS(T), dependent on the evaluating technologies T:

$$\text{PEAKS}(T) = \bigcup_{t \in (T \cap PEAKS)} t. \tag{9.35}$$

The qualitative value for *Peaks* is then computed as per following Equation:

$$\begin{aligned} &\text{QPeaks}(T) = \\ &= \begin{cases} \text{yes} & \text{if } |\text{PEAKS}(T)| > 1 \\ \{\text{yes}/0.5,\ \text{no}/0.5\} & \text{if } |\text{PEAKS}(T)| = 1 \text{ and PEAKS}(T) = \{hydro\} \\ \text{yes} & \text{if } |\text{PEAKS}(T)| = 1 \\ \text{no} & \text{otherwise} \end{cases}. \end{aligned} \tag{9.36}$$

Here note that the second line introduces a qualitative distribution value. In this study, this is one of the sources of probabilities that will be achieved in the end evaluations of technology mixtures. The second source of probabilities are the relational aggregation functions, where each aggregation function will consider the weight of respective technology. Generally, the relational aggregation functions will produce probabilistic distributions, which will be considered during evaluations of higher aggregation functions.

Values for *Energy supply to all*, *Protection of vulnerable groups* and *Lifetime of supply* are defined by the initial beginning state *year*, s_{2013} and the scenario S. Each event of the scenario may increase the value of each attribute by one, or decrease it by one. Furthermore, the year of the evaluation is important during the simulation, because the increase/decrease takes effect only on years, the event is taken into effect. If the simulation year is lower than the event is going to happen, the increase/decrease is not done. For example, the concrete initial values for the attributes in question, are given for the state of the technologies in Slovenia, in year 2013.

The initial value of *Energy supply to all* is *high*. Events that increase the value of the attribute are: JEK1 not happening, JEK2, HESM and HESL. Event that decreases the value of the attribute by one is JEK1. The function with such properties is not presented, however we assume that it is implemented as QEnergy supply to all($initial, T, y$).

Initially, the value for *Protection of vulnerable groups* is *{low/0.2, medium/0.8}*. Here, events influence values in the distribution. If an event may increase the value of the attribute, then each value in the distributions is increased by one. Similarly, events reducing the value reduce each value by one. Here note that the distribution may converge to a single value of *low* or *high*. Events that may increase the value of the attribute are: JEK1 not happening, JEK2, HESM, HESL and GAS. Event that decreases the value of the attribute by one is JEK1. We assume the function with such properties is implemented as QProtection of vulnerable groups($initial, T, y$).

The final input attribute's value *Lifetime of supply* is initially set to *short*. Events that increase the value of the attribute by one are: JEK1 not happening and JEK2. Event that decreases the value of the attribute is JEK1. Again, we assume that such function is implemented as QLifetime of supply($initial, T, y$).

9.3.2 Simulation Model

The simulation model was programmed in Java programming language for the purpose of this study only, not using any existing framework or methodology. Here, we present the pseudo code of the simulation model. The simulation expects the initial state of the system in 2013 (represented by the present technologies, installed power and energy produced), a list of possible events, and the upper limit for the year of simulation. The lower limit for the year is predefined to be 2013. The simulation expects that functions ΔPOWER_X and ΔENERGY_X are available for all events. The simulation also assumes that input values for all supported technologies are available during creation of relational alternatives.

The pseudo code for the simulation is presented in Algorithm 9.2. The simulation iterates through all possible scenarios (2^n, where n is number of events), and then through all years from *Byear* (year to begin the simulation) to the user defined end year, *Eyear*. Now, given the *scenario* and *year*, it gets all available technologies at given *year* with assumed *scenario*, via function *getAlternatives*. The function computes the plausibly existing technologies according to the initial present technologies, and modifies the list according to technologies which might appear according to *scenario*. Then, it creates the main alternative *techMix*, and attaches the relational alternatives *technology*. Finally, it assigns the qualitative values to the regular inputs of the model M, using function *fillMixValues* (presented in Algorithm 9.3), and writes the produced evaluation to the

$OUTPUT$ variable.

Algorithm 9.2: Algorithm represents the running of the simulation from user defined year onwards. Given the initial technologies, their respective installed power and produced energy, events, beginning year, end year, and model for technology mixture assessment, the simulation outputs all possible outcomes of the system from beginning year until the desired end year, considering all combinations of occurring events.

Input: Initial technologies $T = (t_1, t_2, \ldots, t_n)$,
Installed powers $P = (p_1, p_2, \ldots, p_n)$,
Produced energies $EN = (en_1, en_2, \ldots, en_n)$,
Events $EV = (e_1, e_2, \ldots, e_m)$,
Beginning year, $Byear$,
End year, $Eyear$,
Technology mixture assessment model M,
Initial value for *Energy supply to all*, ESA,
Initial value for *Protection of vulnerable groups*, PVG,
Initial value for *Lifetime of supply*, LS
Output: Evaluations of all combinations of events through years $Byear$ — $Eyear$

```
1  if Byear < 2013 then          /* Beginning year must be higher than 2012 */
2  |  return error;
3  m = |EV|;
4  OUTPUT = array [2^m][Eyear − Byear + 1];
5  for scenario = 0; scenario < 2^m; i = i + 1 do          /* For each scenario */
6  |  for year = Byear; year ≤ Eyear; year = year + 1 do    /* For each year */
7  |  |  technologies = getAlternatives(T, scenario, year);
8  |  |  if technologies == {} then          /* No relational alternatives */
9  |  |  |  OUTPUT[scenario][year] = error;
10 |  |  |  continue;
11 |  |  techMix = M.emptyAlternative();
12 |  |  for technology ∈ technologies do
13 |  |  |  techMix.addRelationalAlternative(technology);
14 |  |  end
15 |  |  fillMixValues(techMix, technologies, P, EN, scenario, year, ESA, PVG, LS);
16 |  |  OUTPUT[scenario][year] = M.evaluate(techMix);
17 |  end
18 end
19 return OUTPUT;
```

The Algorithm 9.3 presents the pseudo code for the $fillMixValues$ function. It expects the main technology mixture alternative, technologies that are present in the mixture (already modified by the *scenario*), installed powers and produced energies of the original technologies, the *scenario* which guides the simulation, *year* of evaluation, and initial input values for *Energy supply to all*, *Protection of vulnerable groups* and *Lifetime of supply*. The algorithm does not return a result, however it computes and assigns the qualitative values to regular inputs, and also computes the respective *Weight*'s of the technologies. Firstly, the algorithm computes the *powers* and produced *energies* of the considered technologies, modified by the *scenario*, according to functions ΔPOWER and ΔENERGY. This is done in lines 13 through 15. The actual powers and installed capacities are assigned in lines 4 through 6. Here, function $indexOfTechnology$ gets the index of technology in array T,

from the positional index in array of installed powers. Furthermore, lines 9 through 15 modify the installed capacities and produced energies with ΔPOWER and ΔENERGY. Function *getTechnology* produces the index of influenced technology in array T, by the event i. Also note, the 10-th line of code checks if the i-th bit in the *scenario* is set — if it is, the i-th event is going to happen, otherwise it is not going to happen in this scenario. Next, the algorithm assigns a value to $Weight$ attribute, to the respective technology — based on the proportion of installed power by a particular technology. Finally, it assigns qualitative values to the regular input attributes of technology mixture, via respective functions presented in Section 9.3.1.

9.4 Evaluation of Alternatives

The evaluation of alternatives was done in three stages. The first stage was to evaluate all basic technologies with model T — hydro, coal fired, oil fired, gas fired, nuclear, biomass fired, photovoltaic and wind. After that, the experts B. Kontić et al. (2014b) designed seven technology mixtures, which were evaluated with model M. The technology mixtures represent some situations that were interesting from the expert's point of view. The final stage was running the simulation with the initial state, which represents the state of the electric technologies in Slovenia in year 2013. The initial state was also one of the technology mixtures defined in the second stage of alternative evaluations.

Table 9.6 gives the input values for the eight considered technologies. The input values for each technology appear in columns. The first row of the table gives the name of the technology, and the first column gives the names of the respective attribute. Furthermore, the values appearing in a column are split three-way — the upper part considers input values to the sub-tree for *Rationality* assessment, the middle part considers inputs for *Feasibility* assessment, and the lower part of the table gives the input values for attributes belonging to *Uncertainties* assessment. The basic input values in this case are either crisp or interval values over the domain of the respective input attribute.

Each technology represents an alternative which can be evaluated with model T. Here note that Table 9.6 is missing the input value for the *Weight* attribute. This is due to the fact that the *Weight* does not influence evaluation of single technologies. The evaluations of the technologies present the general overall sustainability assessment for the particular technology, independent of the actual implementation.

For the second stage of alternative evaluation, the experts in the study designed seven technology mixtures, which were evaluated with model M. The different technology mixtures represent seven different most interesting scenarios that may occur in the future:

- M0 represents a technology mixture, which describes the initial state of technologies in year 2013, in Slovenia.
- M1 is a technology mixture, describing a situation without nuclear power and without gas technologies, in year 2030.
- M2 represents a state of technologies in year 2025 if no unplanned changes are made.
- M3 represents a state of technologies in year 2050 if no unplanned changes are made.
- M4 is representing a technology mixture in year 2030, without any nuclear power.
- M5 is a technology mixture in year 2030, considering we do not build Unit 2 of Nuclear Power Plant Krško.

Algorithm 9.3: Algorithm assigns the qualitative input values to the technology mixture, according to the installed powers, produced energy, currently considered scenario and the year of evaluation. It also computes and assigns the respective weights of the considered technologies.

Input: Technology mixture alternative $techMix$,
Technologies $T = (t_1, t_2, \ldots, t_m)$,
Installed powers $P = (p_1, p_2, \ldots, p_n)$,
Produced energy $EN = (en_1, en_2, \ldots, en_n)$,
Current considered *scenario*,
Current *year* of evaluation,
Initial value for *Energy supply to all*, ESA,
Initial value for *Protection of vulnerable groups*, PVG,
Initial value for *Lifetime of supply*, LS

```
1  powers = array [|T|];
2  energies = array [|T|];
3  for i = 0; i < |P|; i = i + 1 do
4  |  j = indexOfTechnology(T, i);
5  |  powers[j] = powers[j] + P[i];
6  |  energies[j] = energies[j] + EN[i];
7  end
8  for i = 0; i < log2(scenario); i = i + 1 do     /* For each event in scenario */
9  |  j = getTechnology(i);
10 |  if (scenario ≫ i)&&1 == 1 then
11 |  |  powers[j] = powers[j] + ΔPOWER_i(year, True);
12 |  |  energies[j] = energies[j] + ΔENERGY_i(year, True);
13 |  else
14 |  |  powers[j] = powers[j] + ΔPOWER_i(year, False);
15 |  |  energies[j] = energies[j] + ΔENERGY_i(year, False);
16 end
17 sumP = sum(powers);
18 sumEN = sum(energies);
19 for i = 0; i < |T|; i = i + 1 do
20 |  T[i].Weight = powers[i]/sumP;
21 end
22 techMix.Installed capacity = QInstalled capacity(T, scenario, year);
23 techMix.Energy produced = QEnergy produced(T, scenario, year);
24 techMix.Base load = QBase load(T, scenario, year);
25 techMix.Peaks = QPeaks(T);
26 techMix.Energy supply to all = QEnergy supply to all(ESA, T, year);
27 techMix.Protection of vulnerable groups = QProtection of vulnerable groups(PVG,
   T, year);
28 techMix.Lifetime of supply = QLifetime of supply(LS, T, year);
```

- M6 is a technology mixture in year 2030, considering we turn down Unit 1 of Nuclear Power Plant Krško.

Table 9.7 gives the values needed to create an alternative for the technology mixture. Each technology mixture is described by the installed powers (given in MW) and produced energies (given in GWh) of all technologies, with additional values for basic input attributes

Table 9.6: The input values for technologies that are supplied into the model for technology sustainability assessment. The table gives a crisp or an interval qualitative value for each input attribute of model T. The input attributes are listed in the left-most column, separated by the sub-tree of the model (*Rationality*, *Feasibility* and *Uncertainties*). The input values for each of the eight technologies are given to the right.

	Hydro	Coal	Oil	Gas	Nuclear	Biomass	Photovoltaic	Wind
Economic	high	high	[medium, high]	high	high	medium	low	low
Societal	[medium, high]	high	medium	high	high	medium	[low, med]	low
Technical level	medium	high	medium	high	high	medium	high	high
Expected development	[low, medium]	[medium, high]	[low, medium]	[low, medium]	high	low	[medium, high]	high
Energy price	[medium, low]	[high, medium]	high	medium	low	high	high	high
Financial sources	certain	certain	less certain	certain	certain	[uncertain, less certain]	[less certain, certain]	[less certain, certain]
Financial shares	more suitable	more suitable	suitable	suitable	more suitable	less suitable	less suitable	less suitable
Long-term liabilities	more suitable	suitable	suitable	less suitable	suitable	less suitable	less suitable	less suitable
Energy ratio	[medium, high]	high	medium	high	high	medium	low	low
Return period	[medium, short]	short	medium	[medium, short]	short	[long, medium]	long	[medium, short]
Dependence	[high, none]	low	[very high, medium]	[very high, medium]	[very high, low]	medium	very high	very high
Land availability	[medium, high]	[medium, high]	high	high	high	low	medium	[low, medium]
Energy share provision	low	medium	[low, medium]	[medium, high]	high	low	low	low
Water protection	[weak, effective]	[weak, present]	present	effective	present	present	effective	effective
Land protection	[present, effective]	[weak, present]	effective	effective	effective	[weak, present]	[weak, present]	[weak, present]
Landscape protection	[present, effective]	[weak, present]	effective	effective	effective	[weak, present]	[weak, present]	[weak, present]
Pollution	low	high	medium	low	low	medium	low	low
Greenhouse gases	low	high	high	high	low	low	low	low
Other pollutants	low	high	medium	medium	low	medium	low	low
Technological complexity	more suitable	more suitable	suitable	suitable	more suitable	suitable	[less suitable, suitable]	[less suitable, suitable]
Infrastructure availability	high	high	high	high	high	[low, medium]	[low, medium]	low
Fuel availability	high	high	medium	medium	high	high	high	high
Fuel accessibility	high	high	medium	medium	high	low	high	high
Investment feasibility	high	medium	medium	medium	high	[low, medium]	low	low
Return of investment	[suitable, more suitable]	[suitable, more suitable]	suitable	suitable	[suitable, more suitable]	suitable	[less suitable, suitable]	[less suitable, suitable]
Social acceptance	[medium, high]	[medium, high]	medium	[medium, high]	[low, high]	[low, medium]	[low, high]	[low, high]
Permitting	yes	yes	yes	yes	yes	yes	yes	yes
Spatial suitability	high	high	high	high	high	high	high	high
Contracts	no rest.	mod. rest.	mod. rest.	mod. rest.	[strong rest., mod. rest.]	mod. rest.	mod. rest.	mod. rest.
Special materials	no rest.	[mod. rest., no rest.]	[mod. rest., no rest.]	[mod. rest., no rest.]	[strong rest., mod. rest.]	[mod. rest., no rest.]	mod. rest.	mod. rest.
Weather dependence	[high, medium]	low	low	low	low	medium	high	high
Fuel supply dependence	[medium, low]	low	[high, medium]	[high, medium]	[high, low]	[medium, low]	low	low
Political stability	high	high	low	low	high	high	high	high
Possible societal changes	positive	positive	no	no	positive	no	no	no
Possible world changes	no	positive	no	no	no	no	no	no
Perception of risks	[medium, none]	[medium, low]	none	[high, medium]	[very high, low]	none	low	none

Energy supply to all, *Protection of vulnerable groups* and *Lifetime of supply*. Here note, if the respective technology's installed power is 0, then the technology is not present in the technology mixture. For each technology mixture, an alternative for model M is created, and for each technology in the technology mixture a relational alternative is added. The relational alternative is created with basic input values presented in the respective column of Table 9.6. Basic input attributes of model M are populated according to the procedures described in Section 9.3.1, or by the assigned input values presented in Table 9.7. Here we see, the basic inputs *Protection of vulnerable groups* values defined in the table, together with proportions of each technology's installed power generate alternatives that will in general have a distributed evaluation (a probabilistic value distribution).

Figure 9.2 shows the evaluation scheme for evaluating technology mixtures *M0*. Respective relational alternatives (technologies HYD — Hydro Power, COA — Coal Power, NUC — Nuclear Power, and PV — Photovoltaic Power) are evaluated using model T. Relational aggregation functions produce input values to model M, which together with input values of alternative *M0* supply all needed input values for model M. With that, model M produces the evaluation of technology mixture *M0* — *unsuitable*.

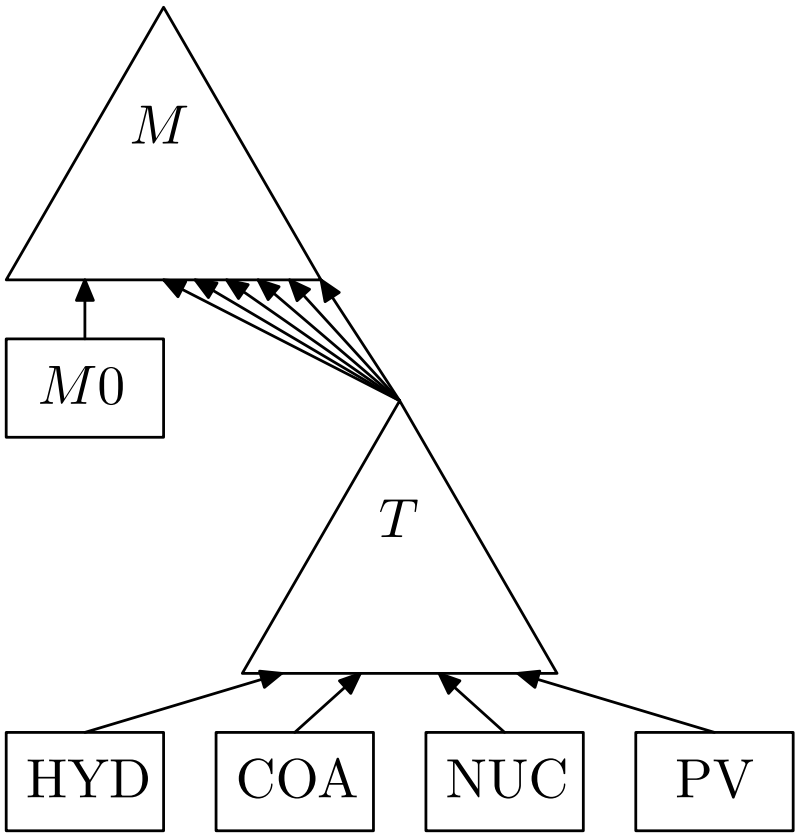

Figure 9.2: Evaluation of technology mixture *M0* proceeds by firstly evaluating technologies in the technology mixture (HYD — Hydro Power, COA — Coal Power, NUC — Nuclear Power, and PV — Photovoltaic Power) with model T. Then, respective relational aggregation functions produce input values to model M (together with input values of *M0*), which are used to produce the final evaluation of *M0*.

The third stage of alternative evaluation is the running of the simulation model S, and evaluating alternatives in the course of the simulation. The simulation needs ten inputs for it to run successfully: initial installed technologies, installed powers, produced

Table 9.7: The seven interesting technology mixtures defined by the expert in the study. Each technology mixture is described by each single technology's installed power and energy produced, and values for attributes of *Energy supply to all*, *Protection of vulnerable groups* and *Lifetime of supply*.

	M0	*M1*	*M2*	*M3*	*M4*	*M5*	*M6*
Hydro Power	1,200	1,400	1,600	1,400	1,400	1,400	1,400
Hydro Energy	4,088	4,760	5,450	4,760	4,760	4,760	4,760
Coal Power	1,000	1,000	1,450	1,000	1,000	1,000	1,000
Coal Energy	4,295	4,800	6,960	4,800	4,800	4,800	4,800
Oil Power	0	300	300	300	300	300	300
Oil Energy	0	0	0	0	0	0	0
Gas Power	0	0	300	600	600	600	600
Gas Energy	0	0	1,500	3,000	3,000	3,000	3,000
Nuclear Power	700	0	2,300	1,600	0	700	1,600
Nuclear Energy	2,512	0	14,000	11,520	0	2,512	11,520
Biomass Power	0	200	50	200	200	200	200
Biomass Energy	0	600	150	600	600	600	600
Photovoltaic Power	250	2,000	500	2,000	2,000	2,000	2,000
Photovoltaic Energy	250	2,000	500	2,000	2,000	2,000	2,000
Wind Power	0	400	100	400	400	400	400
Wind Energy	0	800	200	800	800	800	800
Energy supply to all	high	low	high	high	low	high	medium
Protection of vulnerable groups	{low/0.2, medium/0.8}	{low/0.25, medium/0.75}	{low/0.1, medium/0.5, high/0.4}	{low/0.1, medium/0.5, high/0.4}	{low/0.25, medium/0.75}	{low/0.2, medium/0.8}	{low/0.1, medium/0.5, high/0.4}
Lifetime of supply	short	short	long	long	short	medium	long

energies, considered events, beginning year, end year for the simulation, technology mixture assessment model, and initial values for *Energy supply to all*, *Protection of vulnerable groups* and *Lifetime of supply*. The first three arguments and the last three arguments are derived from the initial state of technologies in year 2013 — technology mixture *M0*. The events used in the simulation are the events TEŠ1, JEK1, JEK2, HESM, HESL, GAS. Beginning year is 2013 and end year for the simulation is set to 2050. The technology mixture assessment model is the actual model M.

9.5 Results

The results discussion, as the evaluation of alternatives, is done in three stages.

9.5.1 Results for Individual Technologies

In the first stage, the eight considered technologies were evaluated using model T. The model identified three main technologies that are most suitable in Slovenia: hydro, gas, and nuclear. The evaluations of the technologies are as follows:

1. Hydro technology was evaluated with an interval value from *suitable* to *excellent*. The suitable assessment came from the fact that *Rationality* may be *low* and *Uncertainties* may be *high*. These evaluations come from the fact that sub-attributes are assessed with lower values. On the other hand, the attributes may have a higher value (note the input values are interval values), and the end assessment may indeed be *excellent*.

2. Coal technology is assessed as *unsuitable*. The worst value is acquired from the fact that the *Rationality* is inappropriate because the *Land use and pollution* is *unsuitable*.

3. Oil technology is again assessed as *unsuitable*. Here the *Rationality* is *unsuitable* and also the evaluation of *Uncertainties* is an interval from *very high* to *low*. Even though the input values here are intervals, the aggregation functions in the end produce a crisp end result.

4. Gas technology has an interval evaluation, from *weak* to *good*. Even though the *Rationality* is assessed as *high*, and also *Feasibility* is *high*, the main factor for lower evaluation is the value for *Uncertainties* — these are assessed as an interval value from *very high* to *medium*.

5. Nuclear technology was also assessed as an interval from *weak* to *excellent*. Again, the *Rationality* and *Feasibility* is assessed to a perfect value, but the evaluation of *Uncertainties* (interval from *very high* to *low*) results in a not perfect evaluation.

6. Biomass technology is assessed as a crisp value of *unsuitable*. Here, *Rationality* is *inappropriate* (mainly due to the bad *Economy* and *Land use and pollution*) and the *Feasibility* is evaluated as an interval from *low* to *medium*. On the positive side, the *Uncertainties* are *low*.

7. Photovoltaic technology is assessed as *unsuitable*. The technology is assessed as *inappropriate* to *low* in terms of *Rationality* (due to bad evaluation of *Economy*), *low* in terms of *Feasibility* and *very high* in terms of *Uncertainties*.

8. Wind technology is also assessed as *unsuitable*. The *Rationality* is *inappropriate* because of poor performance of *Economy* and *Land use and pollution*. *Feasibility* and *Uncertainties* are both given the worst value, *low* and *very high*, respectively.

From these results, we can indeed confirm that *hydro*, *gas* and *nuclear* technologies are all appropriate from the sustainability point of view for Slovenia. Coal, oil and biomass energy sources suffer a bad evaluation mainly due to the bad evaluations of *Rationality*. The renewable energy sources (photovoltaic and wind) were found to be limited in all three factors for sustainability assessment (*Rationality*, *Feasibility* and *Uncertainties*).

9.5.2 Results for Technology Mixtures

The second stage are the evaluations of technology mixtures *M0-M6* with model M. The evaluations of technology mixtures are presented graphically in Figure 9.3. Each bar of the figure corresponds to a particular technology mixture. The height of a specific color's bar is the respective probability for that value. The colors are arranged from worst according to the preference over values of $D_{\text{Technology mixture}}$. The red colored bars represent the worst values (*unsuitable*) and the green colored bars represent the relative probabilities of the best values (*excellent*).

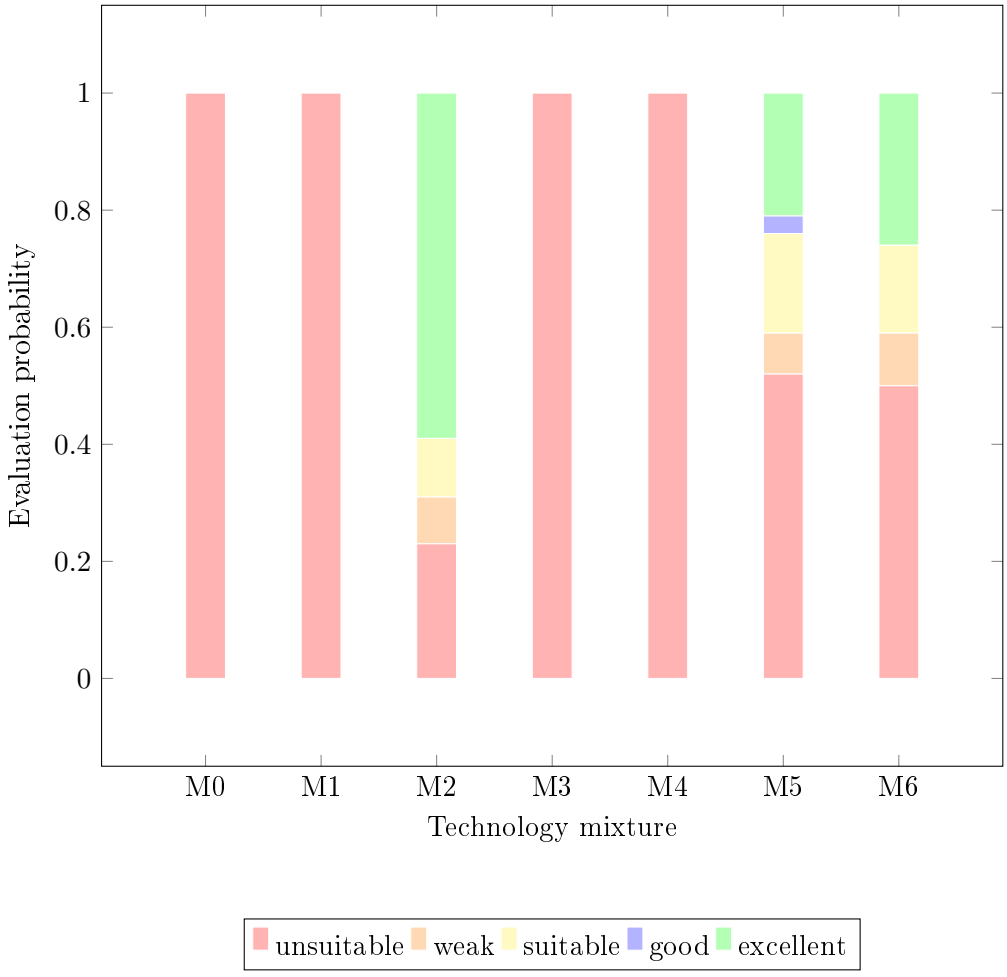

Figure 9.3: The evaluations of the technology mixtures. Each bar corresponds to a technology mixture. On the left-hand side is the probability axis. The height of each color for a technology mixture denotes the probability of the particular value.

The figure shows that technology mixtures *M0*, *M1*, *M3* and *M4* all received an *unsuitable* evaluation. In all cases the values for *Reasonability* and values for *Long-term appropriateness* were *unreasonable* and *low*, respectively. Except in the case of *M3*, where

the evaluation of *Long-term appropriateness* was distributed among *low* with probability 0.19 and *high* with probability 0.81 — the value distribution was {low/0.19, high/0.81}. However, the higher probability in the evaluation of *Long-term appropriateness* did not result in the higher evaluation of *M3*.

The technology mixtures *M2*, *M5* and *M6* evaluated to distributed values, displayed in the respective rows of the second column of Table 9.8. These technology mixtures all have distributed values for evaluations of *Reasonability* and *Long-term appropriateness*. In the table, we also give the values needed for computing the stochastic dominance. The probabilities for all $w \in D_{\text{Technology mixture}}$, $P[E \geq w]$ are given in columns 3 through 7. The probabilities are computed according to Equation (4.22). From the equations, we see that technology mixture *M2* stochastically dominates all other technology mixtures — for every other technology mixture *MX* it holds $P[M2 \geq w] \geq P[MX \geq w]$, for all $w \in D_{\text{Technology mixture}}$ and $P[M2 \geq \text{excellent}] > P[MX > \text{excellent}]$. Similarly it can be shown that *M6* stochastically dominates all technology mixtures, except *M2*, and also that technology mixture *M5* stochastically dominates all mixtures, except *M2* and *M6*. Furthermore, the technology mixtures *M0*, *M1*, *M3* and *M4* are stochastically incomparable.

9.5.3 Simulation Results

The third stage is the running of the simulation model S. We used the initial state as previously defined technology mixture *M0*. We considered the following six events: TEŠ5, JEK1, JEK2, HESM, HESL and GAS. The simulation constructs one alternative (technology mixture) for each year and scenario pair. Since the evaluation years are 2013–2050 and there are $2^6 = 64$ scenarios, there are 2432 technology mixtures. Considering all technology mixtures, their sum of number of relational alternatives is 10080.

The running of the simulation takes about 25 seconds on a fairly modern computer. The simulation produces raw output of all input values and aggregated values for all relational alternatives and main alternatives. Additionally, the procedure uses the *createWebPage* function, provided by DEXx library, to produce a web page. The web page graphically shows the evaluation results, input values and generated relational alternatives. We developed a Decision Support System *(DSS)*, for condensed analysis and overview of different scenarios. The DSS is presented in the following Section 9.6.

9.6 Decision Support System

We developed a Decision Support System *(DSS)* for analysis of different scenarios through time, using the simulation model S. During the running of the simulation model, the procedure outputs raw files, which are later consumed by a web page. The DSS is publicly available to general public (B. Kontić et al., 2014a). The DSS page enables the users to select the events that may happen (defining a scenario), and the evaluation results are dynamically updated according to selected events. With this we enabled the users to see, how the sustainability assessment of electric energy production technologies changes through time. Additionally, we also plot a chart showing the scenario's energy production through time, and the forecasted energy consumption.

Figure 9.4 presents the DSS web page. On the top, DSS gives the user an option to select the events that will occur. Following are three charts that are updated dynamically, based on the selected events. They show time-series data for different measures. The first chart plots the requirements for the energy (*Base needs* and *Additional* needs, in blue and green, respectively) and the actual *Energy produced* (with yellow color) by the technology mixture in a respective year. The second chart displays a time-series of evaluation distributions

Table 9.8: The evaluations of technology mixtures along with the probabilities used for acquiring stochastic dominance.

Technology mixture	evaluation (E)	$P[E \geq \text{unsuitable}]$	$P[E \geq \text{weak}]$	$P[E \geq \text{suitable}]$	$P[E \geq \text{good}]$	$P[E \geq \text{excellent}]$
M0	{unsuitable/1}	1	0	0	0	0
M1	{unsuitable/1}	1	0	0	0	0
M2	{unsuitable/0.23, weak/0.08, suitable/0.1, excellent/0.59}	1	0.77	0.69	0.59	0.59
M3	{unsuitable/1}	1	0	0	0	0
M4	{unsuitable/1}	1	0	0	0	0
M5	{unsuitable/0.52, weak/0.07, suitable/0.17, good/0.03, excellent/0.21}	1	0.48	0.41	0.24	0.21
M6	{unsuitable/0.5, weak/0.09, suitable/0.15, excellent/0.26}	1	0.5	0.41	0.26	0.26

through years 2013–2050. In red, the probability of *unsuitable* is displayed. Similarly, probabilities for *weak*, *suitable*, *good* and *excellent* are displayed in orange, yellow, blue and green color, respectively. The third chart shows the expected value of the distribution displayed in the second chart, through years 2013 to 2050. Concretely, the figure shows the charts when events TEŠ5, JEK2, HESM, HESL and GAS will occur.

9.7 Concluding Remarks

The purpose of the study was to develop a model, which makes transparent and reproducible identification of reliable, rational, and environmentally sound electric energy production technologies in Slovenia by 2050 (B. Kontić, Bohanec, Trdin, et al., 2014; B. Kontić et al., 2014b; Bohanec et al., 2015a, 2015b; B. Kontić et al., 2016). The model was partially developed in DEXi (Bohanec, 2015b), and finalized in the DEXx framework. The purpose of the use-case was to show the development and evaluation of the study using DEXx framework — using native hierarchies, relational aggregation functions and numeric attributes. Since the model was entirely developed in the framework of this study, there were no *previous* results, to compare the computed results with.

- The sustainability assessment of energy options model can be natively implemented in DEXx framework. However, since DEXx does not provide any means for simulations, the simulation model cannot be natively implemented — the library, however, does provide the ability to use it in a simulation environment.
- Since the model for sustainability assessment of energy options uses hierarchies, numeric attributes and relational alternatives, the implementation of the model would not be natively possible without DEXx.
- The model was initially developed in DEXx, so the final evaluation results of the initial study are not available for comparison. However, the correctness of the evaluation procedure and computed results were carefully verified and tested by experts. Some selected alternatives were evaluated by hand and with the DEXx implementation of the model, and yielded identical results.
- There was no previous study with which we could compare the evaluation time. However, the time of 25 seconds for running the whole simulation procedure is acceptable, considering the size of the model and input size.

In contrast with the previous two use-cases (presented in Chapters 7 and 8), which revisit previous projects, the model developed in this chapter is novel — it was developed with DEXx from the beginning. Methodologically, it employs all extensions of DEXx (full hierarchies, numeric attributes, probabilistic distributions and relational models), and thus would have been impossible to implement without DEXx. Furthermore, from the applicative viewpoint, this model solves a difficult strategic planning problem in a unique way, and contributes to electric energy production management in Slovenia (B. Kontić et al., 2016) — through the developed online DSS (B. Kontić et al., 2014a). The DSS gives the experts and the general public the ability to see the outcomes of different management decisions instantly, in a graphic form.

As a follow-up to the project described in this chapter, several publications were published in different conferences (B. Kontić, Bohanec, Trdin, et al., 2014; B. Kontić et al., 2014b; Bohanec et al., 2015a, 2015b) and a journal (B. Kontić et al., 2016). Further, three books were published as a work report (B. Kontić, D. Kontić, Zagorc, Matko, et al., 2014) and an additional brochure available to general public (D. Kontić et al., 2014).

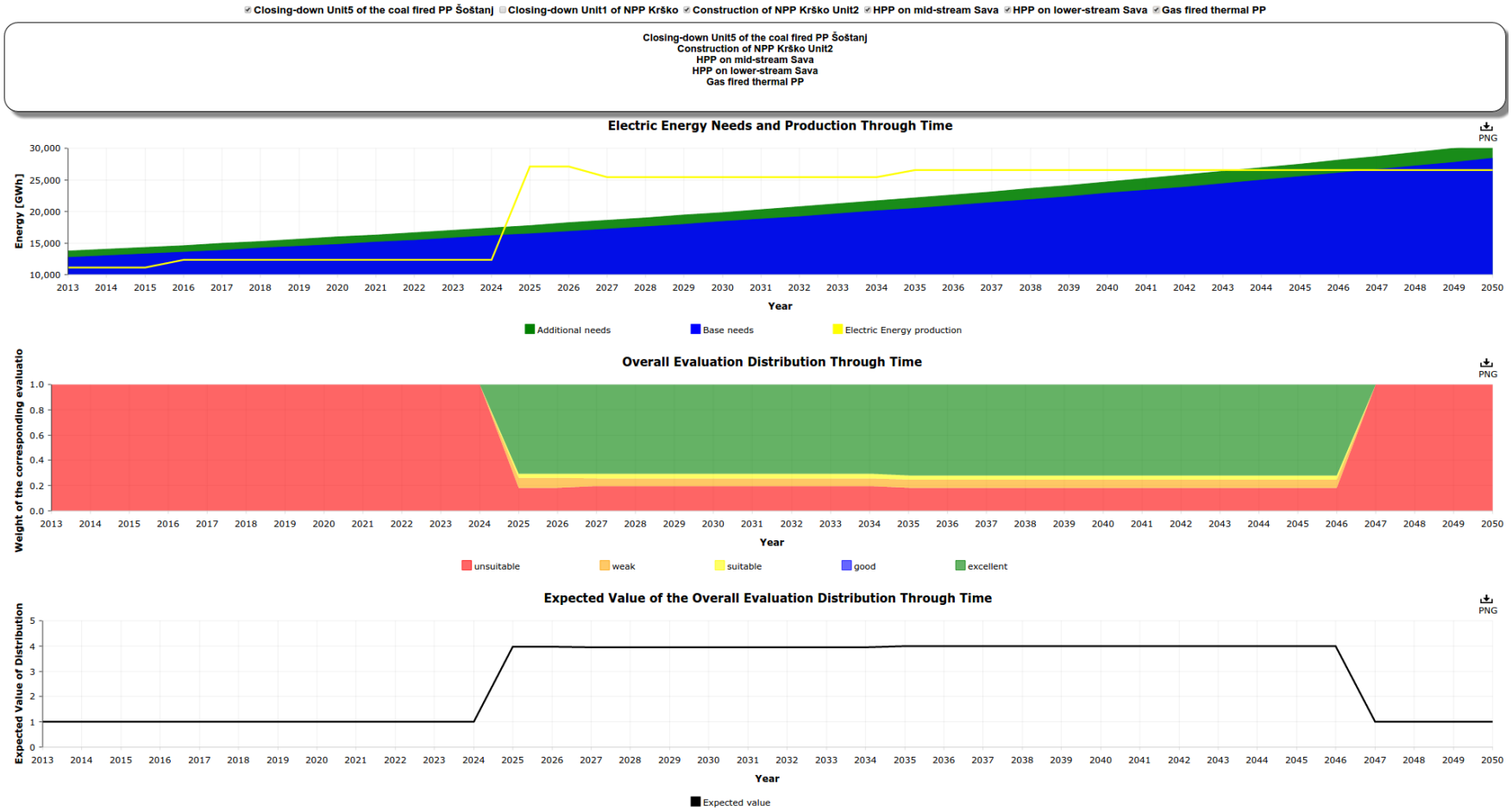

Figure 9.4: Web page showing the results of the evaluation using model S on the beginning state *M0*, and five defined events. The figure shows the option for the user to choose the events that occur and dynamically updates the three charts. The charts represent the base need and additional need for energy, along with energy produced by a scenario in a year. The second graph shows the distribution of final evaluations of scenario through time, and the third chart its expected value.

Chapter 10

Model for Water Flows in Agriculture

This short chapter on modeling water flows in agriculture presents the usage of the DEXx library implementation in a large international industrially-funded project called EVADIFF (Kuzmanovski, 2012; Kuzmanovski et al., 2015; Debeljak et al., 2015). The project is funded by Arvalis (Institut technique au service des agriculteurs et des filières) from France. The particular use-case does not use any extensions to DEX method, presented in this thesis, however, it actively uses the developed implementation of the DEXx library in order to evaluate the developed models in an online application. The online application was developed by authors Debeljak et al. (2015), using *Node.js* for graphical user interface, and the back-end of the application is constructed with *Apache Storm* — a framework for distributed processing of very large data sets. The application is developed to be deployed on multiple processing instances, in order to support fast data processing — concurrent processing of queries from different users of the system.

The methodology for modeling water flows in agriculture consists of two components: data mining components for prediction of various agricultural indicators for water flow management in fields (Kuzmanovski et al., 2015; Debeljak et al., 2015), and a decision support system (*DSS*) used for recommending different actions during management of an agricultural field. Four different water flows from an agricultural field are considered: drainage, infiltration, rapid infiltration and run-off by saturation. The actions which the DSS proposes include recommendation of planting different crops on a field and recommendations on using different herbicides. Recommendations are based on the inputs from the data mining component and the previous states of the particular field (supplied by the user). The decision support component consists of 28 models, developed with DEXi and translated to DEXx, for incorporation in the Apache Storm framework.

The decision support system is implemented as an online application for Aravlis's internal use. The data mining models are run monthly on fresh data from various meteorological sources. These runs are coordinated by different triggers placed in the Apache Storm application. For more information on the data mining component, see (Kuzmanovski et al., 2015).

For computing recommendations for some agricultural field, the user inputs the state of his/her field — information about the location, watering, crop management, soil and field condition, and possible pesticide applications. The decision support system then creates a work-flow of needed queries to the data mining models and evaluations of DEXx models with the provided inputs. The work-flow is described as a topology of jobs (in shape of a directed acyclic graph), consumed by the Apache Storm framework. The jobs are executed on processing nodes, on which the application is deployed. The outputs of jobs

produce recommendations for the selected input values — which pesticides may be used and the dosage recommended, and which crops may be planted on the particular input field. The DEXx library is actively used in the implementation of the model evaluation jobs, because each evaluation of a DEXx model consists of evaluating the input values against the respective model. The evaluation jobs are implemented as Apache Storm processing nodes.

Figure 10.1 shows the main entry window for the online application. Here the user may input information about the location, watering, crop management, soil and field conditions, and possible pesticide applications. By clicking the *Start* button on the lower left, a workflow of jobs is deployed for the processing. Figure 10.2 shows one of the possible reports — *Mitigation Report*. On the left, various recommendations are given, and on the right different water flows from the given field are assessed.

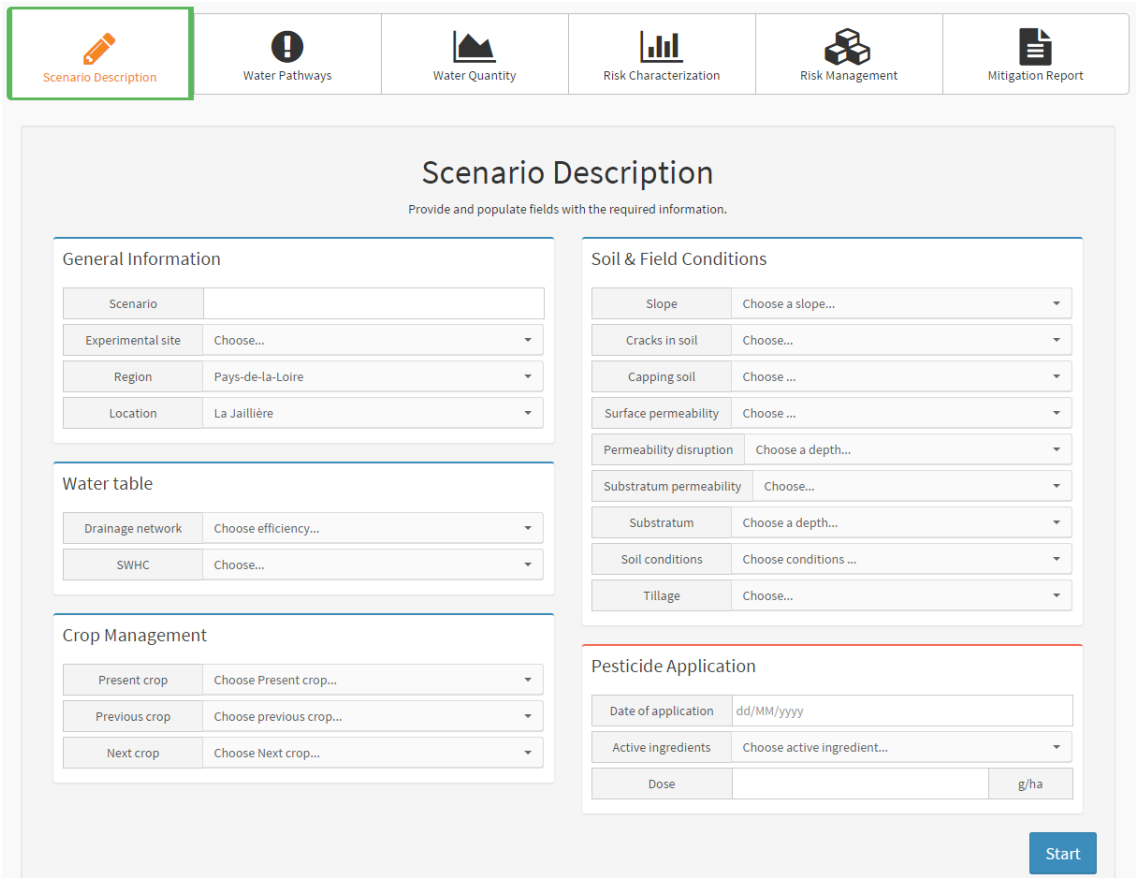

Figure 10.1: The main window of the application developed in the course of EVADIFF project. Here, the user may input information about the location of the field, watering, crop management, soil and field conditions, and pesticide application. The analysis is started by the Start button.

10.1 Concluding Remarks

The purpose of this use-case was to show an advanced application, where the implementation of DEXx is actively used, even though the use-case does not use any of the presented extensions.

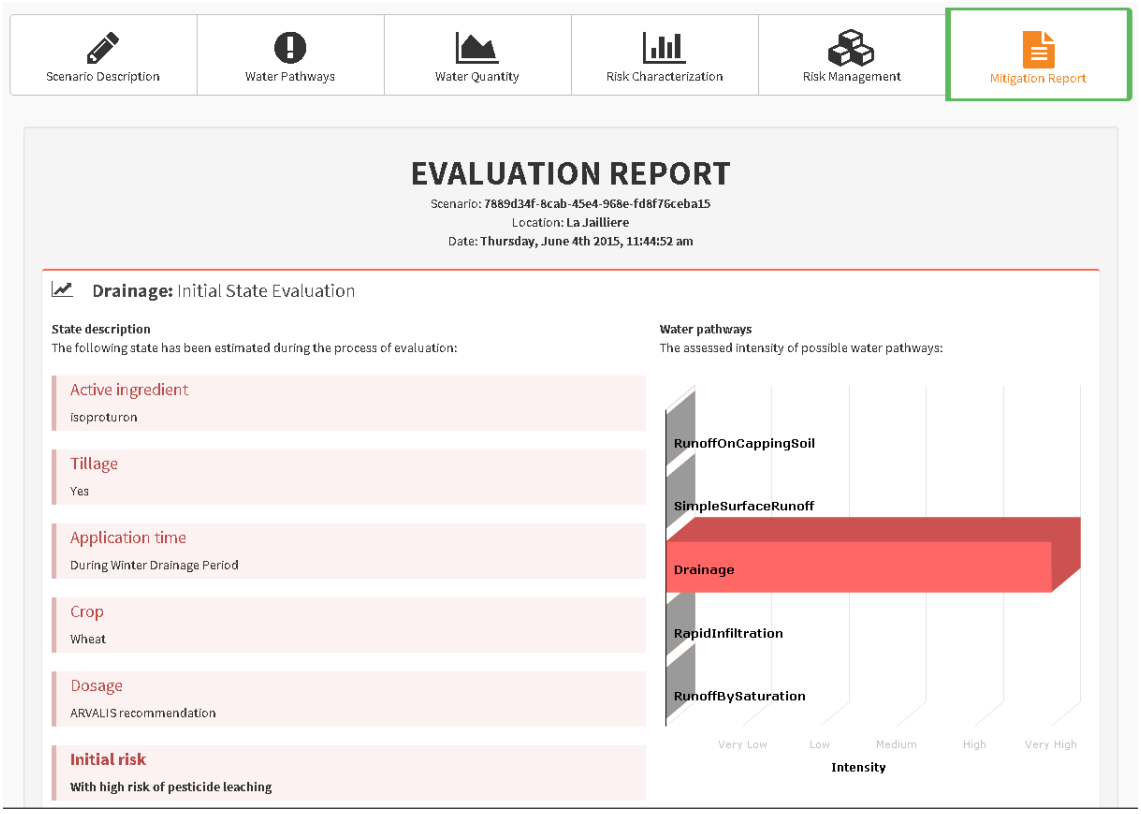

Figure 10.2: The evaluation report is presented to the user, after the running of the jobs. On the left, some general recommendations about the field are given, and on the right, different ways of water flows from the particular field are assessed.

The use-case is assessed with the criteria presented in the methodological part of the introduction, as follows:

- The implementation of the water flows in agriculture model is natively possible using DEXx framework, since DEXx also supports all functionalities provided by DEX.
- Native implementation of the model is also possible, without any of the presented extensions. However, the library implementation of DEXx provides better means to incorporate alternative evaluations into native Java programs. There is an additional tool called *DEXiEval* (Bohanec, 2015b), which must be run from the terminal, and is hence non-native.
- There are no previous results to compare them with the results acquired with the DEXx implementation.
- There are no previous evaluation time results to compare them with the times acquired with the DEXx implementation.

The implementation of the model was, from the initial point of development, meant for usage in a complex environment, where fast data processing must be carried out (using Apache Storm), to serve multiple users at once. Here, DEXx library is crucial to the system, since all DEXx model evaluations are carried out with the developed library — the library is integrated into the system so that each model evaluation is represented as

a processing job. DEXx library has a large impact on the whole system, since the library allows for fast (concurrent) evaluation of models, and its memory footprint is minimal.

The developed online system (together with DEXx library) is actively used by the users from Institut technique au service des agriculteurs et des filières.

Chapter 11

Conclusions

We conclude the dissertation with an overview of the work done, the implied results, consequences, lessons learned, and limitations. Furthermore, we present the ideas for the future work.

11.1 Summary

In this thesis, we introduced five extensions to DEX method: full hierarchies, numeric attributes, general aggregation functions, value distributions, and relational alternatives and models. All extensions were motivated by needs identified when applying DEX in complex real decision modeling tasks, and by increasing demands of the decision problems. The extensions substantially increase the range of decision problems that can be addressed, but they also require a number of changes and additions to the method.

Full hierarchies are now supported with different means of handling hierarchical connections. In order to introduce numeric attributes, it was necessary to add a class of numeric value scales, which in turn increased the number of primitive function types from one to five. The general aggregation functions are subsequently covered with these five primitive function types. The introduction of value distributions required a generalization of value scales to extended domains (probabilistic distributions and fuzzy sets) and an extension of the evaluation algorithm to use existing aggregation functions on value distributions. The new ability to represent and evaluate relational alternatives required the introduction of relational models, relational attributes and relational aggregation functions. All the extensions are compatible with each other so that they can be used simultaneously in one decision making scenario.

Considering the previous applications of DEX and the solutions found in some other MCDA methods, the identified extensions are not entirely new but are for the first time systematically brought together and formalized in the DEX framework. Previously, applications were typically formulated and implemented in an ad-hoc manner, and tailored to a particular application. However, they clearly indicated the need for extending the method — therefore we took them as an inspiration to identify and formalize the extensions. Some of the inspiring applications are: qualitative relational models introduced in the assessment of public administration e-portals (Leben et al., 2006; Leben & Bohanec, 2004) (also see Chapter 7), qualitative probabilistic distributions used in ecological domains (Bohanec et al., 2009, 2008; Bohanec & Žnidaršič, 2008; Žnidaršič et al., 2008), and numeric attributes and relational models in the modeling and assessment of bank reputational risk (Bohanec et al., 2013, 2014) (also see Chapter 8). In this thesis all three extensions were for the first time used together in the evaluation of sustainable electrical energy production in Slovenia (B. Kontić, Bohanec, Trdin, et al., 2014; B. Kontić et al., 2016) (also see Chapter 9).

Method DEX and the extensions were joined into a unified method, called DEXx. During the course of the dissertation we implemented the implementation for DEXx, which facilitates the DM in using the extensions in decision making problems. The library implementation additionally supports several utilities for the DMs, for further viewing and analysis of the developed models and identified alternatives — outside the scope of the DEXx framework.

Finally, we used the DEXx implementation in four use-cases. Two of the use-cases were revisited projects (evaluation of public e-portals and bank reputational risk assessment), which were previously implemented in an ad-hoc manner, and required some degree of user intervention during evaluation. The use-cases for sustainability assessment of electric energy production technologies in Slovenia and decision support system for water flows in agriculture were implemented in DEXx from the beginning. Three of the four use-cases use the implementation of DEXx, and additionally require at least one of the identified extensions. The fourth, decision support system for water flows in agriculture, relies on the implementation of DEXx, however it does not require any of the extensions.

11.2 Contributions

The first and main contribution of this dissertation is the formalization of the already established qualitative multi-criteria modeling method DEX and formalization of five extensions to DEX. Based on identified practical needs in complex decision situations, we included the following extensions to DEX in DEXx:

- Support for *full hierarchies.*
- Inclusion of *numeric attributes.*
- Aggregation of numeric and qualitative values with *general aggregation functions.*
- Evaluation and definition of alternatives with *probabilistic and fuzzy value distributions.*
- Evaluation of *relational alternatives* on *relational models.*

In order to support these concepts, a number of formal and algorithmic components were added, specifically: numeric scale types, extended value domains, probabilistic and fuzzy value distributions, numeric aggregation functions, five primitive function types and relational attributes, relational aggregation functions, relational models and relational alternatives. All these concepts greatly increase the flexibility and representational richness of the method, enlarge the class of decision problems that can be addressed, and improve the decision process by providing additional methods and tools to the decision maker.

The second contribution of this thesis is the design and implementation of a library in Java, called DEXx, which supports the native DEX functionalities, together with all five proposed extensions (see Chapter 6). Even more, the library not only implements the proposed extensions, it also allows for concurrent usage of all extensions in unity, on a single decision model.

The third contribution is construction, application and evaluation of four real-life decision making models, using the implementation of DEXx. It is shown that construction of such decision models is not natively possible without the implementation of extended DEX. The four use-cases are:

- Evaluation of public e-portals (see Chapter 7).

- Assessment of bank reputational risk (see Chapter 8).
- Sustainability assessment of electric energy production technologies in Slovenia (see Chapter 9).
- modeling water flows in agriculture (see Chapter 10).

Among these, the most extensive and difficult is the sustainability assessment of electric energy production technologies in Slovenia. The developed model employs all the developed methodological extensions and contributes to the strategic planning of electric energy production in Slovenia. The model is operational through a decision support system, easily accessible online and available to the general public.

11.3 Assessment of the Extensions

The proposed extensions provide new means for representing DM's knowledge and preferences, and facilitate solving a wider variety of decision problems:

Full hierarchies (see Section 4.1) enable the DM to consider one attribute (or criterion) in multiple parts of the model. In this way, an attribute influences different parts of the model, and is hence included in multiple aggregation functions. Until now, the same functionality was achieved with the so-called links in DEXi. This extension now makes DEXx a native hierarchical methodology.

Numeric attributes and general aggregation functions (see Section 4.2) allow for a more natural treatment of numeric quantities. Instead of being limited only to qualitative attributes, the DM can choose between the qualitative and quantitative representations. This is illustrated on an example in Section 5.1, where numeric attributes can be used for measurable numeric quantities (such as *price*), and qualitative attributes can be used for binary choices, quality judgements and quality estimations (for example *apartment*). Numeric attributes also solve a common problem in DEX: mapping from numeric measurements to qualitative input values. To date, this had to be done implicitly and manually by the DM, whereas DEXx method facilitates this mapping with ease. In principle, the introduction of numeric attributes opens DEX for the inclusion of other approaches of quantitative MCDA, such as pairwise preference and weights elicitation of AHP (Saaty, 2008; Saaty & Vargas, 2012; Saaty, 2005) and using the marginal utility functions of MAUT (Multi-attribute utility theory) (Wang & Zionts, 2008).

For instance, in order to include the AHP method, DEXx can successfully simulate its model structure and alternative evaluation. However, DEXx methodology was not designed to acquire preferential knowledge about alternatives by pairwise comparison as in AHP. While evaluating alternatives in AHP, all must be known in advance for comparison, whereas in DEXx, only the currently evaluating alternative is needed to produce the evaluation.

Value distributions (see Section 4.3) introduce capabilities that are well known and proven in contexts such as expert systems, fuzzy control systems and uncertainty and risk analysis. They allow an explicit formulation of "soft" (imprecise, uncertain and even missing) knowledge and data. Specifically, we introduced two types of value distributions: probabilistic, which are suitable for representing uncertainty, and fuzzy, to represent vague concepts and values. The former resembles the use of probabilities in MAUT (Wang & Zionts, 2008), whereas the latter is in line with

the trend of extending MCDA methods with fuzzy sets (Baracskai & Dörfler, 2003; Kahraman, 2008; Omero et al., 2005).

Even though the usage of value distributions in DEXx models decreases the comprehensibility of the models and increases the development time of such models, it also increases the expressive power of the designed models. Such models can provide evaluations which are not possible to produce using DEXi. Such models can sometimes easily discriminate between close alternatives (via stochastic dominance), where pure qualitative model may not.

Models developed using value distributions will probably require additional input from the original DM, when other users may want to understand it. We assume, the aggregation functions defined using value distributions will not form complex distributions in the edge cases of the inputs (whenever all inputs are best or worst possible). The true power of value distributions becomes evident in the *intermediate* parts of the functions, where value distributions can be used to distinguish close input value situations.

Relational models (see Section 4.4) address decision problems in which alternatives are composed of similar sub-components. In quantitative MCDM, such problems are rarely mentioned because they can be relatively easily handled by common aggregation functions, such as the average or summation. In the qualitative world of DEX, there are no such obvious functions — thus the need for the extension. The key contribution of relational models is to allow the DM to granulate the decision problem to a finer level, explicitly considering parts of the whole.

On the negative side, the extensions increase the complexity of model development, by the following facts:

- Previously in DEX, there was just one conceptual way to define an attribute or an aggregation function — now there are plenty from which the DM can choose. Even though this increases the flexibility of the modeling, it also requires the mastering and understanding of newly available tools. Particularly, the elicitation of attributes and aggregation functions becomes more difficult.

- Additional to dealing with new types of numeric attributes, the DM should also control the interplay between qualitative and numeric attributes, for which there are now five different primitive forms instead of just one (see Section 4.2).

- Aggregation function definition complexity is increased, because of the new classes of numerical functions, which have to be formulated by the DM in accordance with his or her preferences.

- Evaluation of alternatives is more complex, since DEX was previously limited to only qualitative values and their sets, and to evaluation of "flat" alternatives. DEXx additionally produces a wide variety of other value types: integers, real numbers and fuzzy or probabilistic value distributions. Further, the evaluation procedure now considers evaluation of possible relational alternatives. This increases the DM's cognitive load to interpret evaluation results.

- The increased complexity of the models and computed values may degrade the comprehensibility of the method and results for the DM. Consequently, at this stage DEXx seems a more useful tool for a skilled decision analyst than for an ordinary DM.

- Analysis such as ±1 *Analysis* may be harder to perform with included value distributions, and may not produce any results. This is because ±1 *Analysis* relies on the fact that output values are always comparable, whereas this is not true using value distributions, if we are employing stochastic dominance for comparing values. When using stochastic dominance for comparison of value distributions, the result can be undefined.

For the above reasons, the DMs need to gain additional skills to successfully use DEXx. Specifically, the DMs need to successfully discriminate between concepts that should be represented with numeric and qualitative attributes. Further, they need to gain experience and master all five possible primitive aggregation functions types, and how to combine them. To successfully use value distributions, they need to assess if the problem can be modeled with probabilistic distributions or fuzzy sets, and finally understand and interpret the results provided by the evaluation procedure. DMs also need to learn on how to recognize relational decision problems, and model them inside DEXx.

The true contributions of the proposed extensions will become evident with extensive usage in real decision making scenarios, with different DMs feedbacks. Even though generally the modeling time with complex models in DEXx will be longer than with similar models developed in DEXi, the expressive power will provide additional insights to the model and alternatives, which subsequently means the best decision will be easier to identify.

Further research is needed to assess both positive and negative effects, with the aim of finding the best trade-off between the method's richness, flexibility, simplicity and comprehensibility.

11.4 Future Work

We are aware that retrieving knowledge from the DM, especially with methods of such complexity, is a difficult problem. In this thesis, we were not concerned with the difficulties of knowledge acquisition; our primary goal was to open up the DEX method — in a formalized methodological fashion, to make it more suitable for addressing a wide range of real decision problems. We were interested in giving the DM the ability to specify his/her preference as flexibly as possible. We are aware that more research and practice is needed to fully assess the strengths and weaknesses of DEXx method and to identify potential difficulties associated with preference elicitation. In further research, we will attempt to find a suitable balance between the flexibility, simplicity and comprehensibility of the method, which may even require narrowing down its current broadness.

In the future, we aim to implement an interactive computer program that will fully support DEXx method in a way that is similar to the way the current program DEXi supports DEX. We would also like to include more dynamic aspects to the function manipulation, while attribute values are deleted and added. Furthermore, we will implement function manipulation while attributes are added and deleted to the respective aggregated attribute. Also, we would like to implement option analysis for the developed decision models and alternatives, such as, ±1 *Analysis* and displaying alternative evaluations with charts.

References

Alić, I., Bohanec, M., Costante, M., Dinev, V., Foti, M., Grčar, M., ... Sinanaj, G. (2013). FIRST: Highly Scalable Machine Learning and Qualitative Models v2. Retrieved from http://project-first.eu/content/d63-highly-scalable-machine-learning-and-qualitative-models-v2.

Alić, I., Siering, M., & Bohanec, M. (2013). Hot Stock or Not? A Qualitative Multi-Attribute Model to Detect Financial Market Manipulation. In D. L. Wigand (Ed.), *eInnovations: Challenges and Impacts for Individuals, Organizations and Society* (pp. 64–77). Bled, Slovenia: Moderna organizacija.

Arasteh, A., Aliahmadi, A., & Omran, M. M. (2014). A Multi-stage Multi Criteria Model for Portfolio Management. *Arabian Journal for Science and Engineering*, *39*(5), 4269–4283.

Bana e Costa, C. & Vansnick, J.-C. (1999). The MACBETH Approach: Basic Ideas, Software, and an Application. In N. Meskens & M. Roubens (Eds.), *Advances in Decision Analysis* (Vol. 4, pp. 131–157). Netherlands: Kluwer Academic Publishers.

Baracskai, Z. & Dörfler, V. (2003). Automated Fuzzy-Clustering for Doctus Expert System. In *International Conference on Computational Cybernetics*. Siófok, Hungary.

Bede, B. (2012). *Mathematics of Fuzzy Sets and Fuzzy Logic*. Studies in Fuzziness and Soft Computing. Heidelberg, New York, Dordrecht, London: Springer.

Bergez, J.-E. (2013). Using a Genetic Algorithm to Define Worst-Best and Best-Worst Options of a DEXi-type Model: Application to the MASC Model of Cropping-System Sustainability. *Computers and Electronics in Agriculture*, *90*, 93–98.

Bohanec, M. (2014). *DEXi: Program for Multi-Attribute Decision Making: User's Manual, Version 4.01, IJS Report DP-11739*. Jožef Stefan Institute.

Bohanec, M. (2015a). DEX: An Expert System Shell for Multi-Attribute Decision Making. Retrieved from http://kt.ijs.si/MarkoBohanec/dex.html.

Bohanec, M. (2015b). DEXi: A Program for Multi-Attribute Decision Making. Retrieved from http://kt.ijs.si/MarkoBohanec/dexi.html.

Bohanec, M. (2015c). JDEXi: Open-source DEXi Java Library. Retrieved from http://kt.ijs.si/MarkoBohanec/jdexi.html.

Bohanec, M., Aprile, G., Constante, M., Foti, M., & Trdin, N. (2013). Decision Support Model for the Assessment of Bank Reputational Risk. In M. Gams, R. Piltaver, D. Mladenić, M. Grobelnik, F. Novak, B. Blažica, ... A. Brodnik (Eds.), *Information Society — IS 2013* (Vol. A, pp. 11–14). Ljubljana, Slovenia.

Bohanec, M., Aprile, G., Constante, M., Foti, M., & Trdin, N. (2014). A Hierarchical Multi-Attribute Model for Bank Reputational Risk Assessment. In G. Phillips-Wren, S. Carlsson, & A. Respício (Eds.), *17th conference for IFIP WG8.3 DSS* (pp. 92–103). Paris, France: IOS Press.

Bohanec, M., Bertheau, Y., Brera, C., Gruden, K., Holst-Jensen, A., Kok, E. J., ... Žnidaršič, M. (2009). The Co-Extra Decision Support System: A Model-Based Inte-

gration of Project Results. In *Co-Extra International Conference* (pp. 63–64). Paris, France.

Bohanec, M., Messéan, A., Scatasta, S., Angevin, F., Griffiths, B., Krogh, P. H., ... Džeroski, S. (2008). A Qualitative Multi-Attribute Model for Economic and Ecological Assessment of Genetically Modified Crops. *Ecological Modelling*, *215*(1-3), 247–261.

Bohanec, M. & Rajkovič, V. (1983). Processes and Tools for Decision Making. In H. G. Sol (Ed.), (Chap. An Expert System for Decision Making, pp. 235–248). North-Holland.

Bohanec, M. & Rajkovič, V. (1990). DEX: An Expert System Shell for Decision Support. *Sistemica*, *1*(1), 145–157.

Bohanec, M. & Rajkovič, V. (1999). Multi-Attribute Decision Modeling: Industrial Applications of DEX. *Informatica*, *23*(1), 487–491.

Bohanec, M., Rajkovič, V., Bratko, I., Zupan, B., & Žnidaršič, M. (2013). DEX Methodology: Three Decades of Qualitative Multi-Attribute Modeling. *Informatica*, *37*(1), 49–54.

Bohanec, M. & Trdin, N. (2014). Qualitative Multi-Attribute Decision Method DEX: Theory and Practice. In *20th Conference of the International Federation of Operational Research Societies* (p. 239). Barcelona, Spain.

Bohanec, M., Trdin, N., & Kontić, B. (2015a). A Qualitative Multi-Criteria Model for the Evaluation of Electric Energy Production Technologies in Slovenia. In L. Zadnik Stirn, J. Žerovnik, M. Kljajić Borštnar, & S. Drobne (Eds.), *Proceedings of the 13th International Symposium on Operational Research SOR'15* (pp. 11–16). Bled, Slovenia: Slovenian Society Informatika, Section for Operational Research.

Bohanec, M., Trdin, N., & Kontić, B. (2015b). Strategic Assesment of Electric Energy Production Technologies in Slovenia Using Qualitative Multi-Attribute and Simulation Methods. In G. Pflug (Ed.), *OR2015 Vienna Conference: International Conference on Operations Research* (pp. 107–108). Vienna, Austria.

Bohanec, M. & Žnidaršič, M. (2008). Supporting Decisions About the Introduction of Genetically Modified Crops. In P. Zaraté, J. P. Belaud, G. Camilleri, & F. Ravat (Eds.), *Collaborative Decision Making: Perspectives and Challenges* (pp. 404–415). Frontiers in Artificial Intelligence and Applications. Amsterdam, Netherlands: IOS Press.

Boose, J. H., Bradshaw, J. M., Koszarek, J. L., & Shema, D. B. (1993). Knowledge Acquisition Techniques for Group Decision Support. *Knowledge Acquisition*, *5*(4), 405–448.

Bouyssou, D., Marchant, T., Pirlot, M., Tsoukiàs, A., & Vincke, P. (2006). *Evaluation and Decision Models with Multiple Criteria*. New York: Springer Verlag.

Caflisch, R. E. (1998). Monte Carlo and quasi-Monte Carlo Methods. *Acta Numerica*, *7*, 1–49.

Clemen, R. T. & Reilly, T. (2001). *Making Hard Decisions with DecisionTools*. Pacific Grove, CA: Duxbury/Thomson Learning.

Craheix, D., Bergez, J.-E., Angevin, F., Bockstaller, C., Bohanec, M., Colomb, B., ... Sadok, W. (2015). Guidelines to Design Models Assessing Agricultural Sustainability, Based Upon Feedbacks from the DEXi Decision Support System. *Agronomy for Sustainable Development*, 1–17. doi:10.1007/s13593-015-0315-0

Debeljak, M., Džeroski, S., Kuzmanovski, V., Marks-Perreau, J., Trajanov, A., & Real, B. (2015). Decision Support Modelling for Environmentally Safe Application of Pesticides Used in Agriculture. In L. Zadnik Stirn, J. Žerovnik, M. Kljajić Borštnar, & S. Drobne (Eds.), *Proceedings of the 13th International Symposium on Operational*

Research SOR'15 (pp. 17–22). Bled, Slovenia: Slovenian Society Informatika, Section for Operational Research.

Dembczyński, K., Greco, S., & Słowiński, R. (2009). Rough Set Approach to Multiple Criteria Classification with Imprecise Evaluations and Assignments. *European Journal of Operational Research*, *198*(2), 626–636.

Demšar, J., Curk, T., Erjavec, A., Gorup, Č., Hočevar, T., Milutinovič, M., ... Zupan, B. (2013). Orange: Data Mining Toolbox in Python. *Journal of Machine Learning Research*, *14*, 2349–2353.

Durbach, I. N. & Stewart, T. J. (2012). Modeling Uncertainty in Multi-Criteria Decision Analysis. *European Journal of Operational Research*, *223*(1), 1–14.

Efstathiou, J. & Rajkovič, V. (1980). Multi-Attribute Decision-Making Using a Fuzzy, Heuristic Approach. *International Journal of Man-Machine Studies*, *12*(2), 141–156.

Feller, W. (1968). *An Introduction to Probability Theory and Its Applications* (3rd ed.). New York, London, Sydney: Wiley.

Figueira, J., Greco, S., & Ehrgott, M. (2005). *Multiple Criteria Decision Analysis: State of the Art Surveys*. Boston, Dordrecht, London: Springer Verlag.

Frank, E., Wang, Y., Inglis, S., Holmes, G., & Witten, I. H. (1998). Using Model Trees for Classification. *Machine Learning*, *32*(1), 63–76.

French, S. (1986). *Decision Theory: An introduction to the Mathematics of Rationality*. Halsted Press.

Gomes, L. F. A. M., Moshkovich, H. M., & Torres, A. (2010). Marketing Decisions in Small Business: How Verbal Decision Analysis can Help. *International Journal of Management and Decision Making*, *11*(1), 19–36.

Greco, S., Matarazzo, B., & Słowiński, R. (2001). Rough Sets Theory for Multicriteria Decision Analysis. *European Journal of Operational Research*, *129*(1), 1–47.

Greco, S., Matarazzo, B., & Słowiński, R. (2002). Rough Sets Methodology for Sorting Problems in Presence of Multiple Attributes and Criteria. *European Journal of Operational Research*, *138*(2), 247–259.

Greco, S., Matarazzo, B., & Słowiński, R. (2010). Dominance-based Rough Set Approach to Decision Under Uncertainty and Time Preference. *Annals of Operations Research*, *176*(1), 41–75.

Hadar, J. & Rusell, W. (1969). Rules for Ordering Uncertain Prospects. *American Exonomic Review*, *59*(1), 25–34.

Hewitt, E. & Stromberg, K. (1965). *Real and Abstract Analysis*. Berlin, Heidelberg, New York: Springer-Verlag.

Holt, J., Leach, A. W., Schrader, G., Petter, F., Macleod, A., Gaag, D. J. v. d., ... Mumford, J. D. (2013). Eliciting and Combining Decision Criteria Using a Limited Palette of Utility Functions and Uncertainty Distributions: Illustrated by Application to Pest Risk Analysis. *Risk Analysis*, *34*(1), 4–16.

Hurson, C. & Ricci-Xella, N. (1998). Operational Tools in the Management of Financial Risks. In C. Zopounidis (Ed.), (Chap. Multicriteria Decision Making and Portfolio Management with Arbitrage Pricing Theory, pp. 31–55). Springer US.

Ishizaka, A. & Nemery, P. (2013). *Multi-Criteria Decision Analysis: Methods and Software*. Chichester: Wiley.

Jacquet-Lagrèze, E. & Siskos, Y. (1982). Assessing a Set of Additive Utility Functions for Multicriteria Decision Making: The UTA method. *European Journal of Operational Research*, *10*(2), 151–164.

Jacquet-Lagrèze, E. & Siskos, Y. (2001). Preference Disaggregation: 20 Years of MCDA Experience. *European Journal of Operational Research*, *130*(2), 233–245.

Jiang, Z., Zhang, H., & Sutherland, J. W. (2011). Development of Multi-Criteria Decision Making Model for Remanufacturing Technology Portfolio Selection. *Journal of Cleaner Production*, *19*(17-18), 1939–1945.

Kahraman, C. (2008). *Fuzzy Multi-Criteria Decision Making*. Springer Optimization and Its Applications. New York: Springer.

Kepner Higgins, C. & Tregoe, B. B. (1997). *The New Rational Manager*. Princeton, New Jersey: Princeton Research Press.

Kontić, B., Bohanec, M., Kontić, D., Trdin, N., & Matko, M. (2016). Improving Appraisal of Sustainability of Energy Options — A View from Slovenia. *Energy Policy*, *90*, 154–171.

Kontić, B., Bohanec, M., Trdin, N., Kontić, D., Zagorc-Kontić, S., & Matko, M. (2014). Comparative Evaluation of Various Energy Options using Qualitative Multi-Attribute Models. In *20th Conference of the International Federation of Operational Research Societies* (p. 239). Barcelona, Spain.

Kontić, B., Kontić, D., Zagorc, S., Marušić, M., Dermol, U., Bohanec, M., & Trdin, N. (2014a). Comparative Model of Energetic Scenarios. Retrieved from http://sepo.ijs.si/naloge/OVJE/energetic_scenario_comparative_model/.

Kontić, B., Kontić, D., Zagorc, S., Marušić, M., Dermol, U., Bohanec, M., & Trdin, N. (2014b). *Ocena Vzdržnosti za Razvoj Energetike v Sloveniji do Leta 2030 s Poudarkom na Jedrski Tehnologiji. Knjiga 1, IJS Report DP-11583*. Jožef Stefan Institute.

Kontić, B., Kontić, D., Zagorc, S., Matko, M., Dermol, U., Bohanec, M., & Trdin, N. (2014). *Ocena Vzdržnosti za Razvoj Energetike v Sloveniji do Leta 2030 s Poudarkom na Jedrski Tehnologiji. Book 1, IJS Report DP-11583*. Jožef Stefan Institute.

Kontić, D., Kontić, B., Marušič, J., Ogrin, D., Polič, M., Kos, D., ... Kontič, V. (2014). *Ocena Vzdržnosti za Razvoj Energetike v Sloveniji do Leta 2030* (1st ed.) (D. Kontić, Ed.). Ljubljana: Institut Jožef Stefan.

Kuzmanovski, V. (2012). *Integration of Expert Knowledge and Predictive Learning: Modelling Water Flows in Agriculture* (MSc thesis, Jožef Stefan International Postgraduate School).

Kuzmanovski, V., Trajanov, A., Leprince, F., Džeroski, S., & Debeljak, M. (2015). Modeling Water Outflow from Tile-drained Agricultural Fields. *Science of The Total Environment*, *505*(1), 390–401.

Landwehr, N., Hall, M., & Eibe, F. (2005). Logistic Model Trees. *Machine Learning*, *59*(1), 161–205.

Larichev, O. I. (2001). Method ZAPROS for Multicriteria Alternatives Ranking and the Problem of Incomparability. *Informatica*, *12*(1), 89–100.

Larichev, O. I. & Moshkovich, H. M. (1994). An Approach to Ordinal Classification Problems. *International Transactions in Operational Research*, *1*(3), 375–385.

Larichev, O. I. & Moshkovich, H. M. (1995). ZAPROS-LM — A Method and System for Ordering Multiattribute Alternatives. *European Journal of Operational Research*, *82*(3), 503–521.

Larichev, O. I. & Moshkovich, H. M. (1997). *Verbal Decision Analysis for Unstructured Problems*. Theory and Decision Library C. Boston: Kluwer Academic Publishers.

Lavrač, N. & Džeroski, S. (1994). *Inductive Logic Programming: Techniques and Applications*. New York: Ellis Horwood.

Leben, A. & Bohanec, M. (2004). E-uprava: Izbrane Razvojne Perspektive. In M. Vintar, J. Grad, D. Lesjak, & C. Bavec (Eds.), (Chap. Vrednotenje Portalov Življenjskih Situacij, pp. 123–140). Faculty of Administration.

Leben, A., Kunstelj, M., Bohanec, M., & Vintar, M. (2006). Evaluating Public Administration e-Portals. *Information Polity — The development of e-government in Central and Eastern Europe (CEE)*, *11*(3, 4), 207–225.

Michie, D. (1995). Machine Learning: ECML-95. In N. Lavrač & S. Wrobel (Eds.), (Chap. Problem Decomposition and the Learning of Skills, Vol. 912, pp. 17–31). Notes in Artificial Intelligence. Springer-Verlag.

Mileva-Boshkoska, B. & Bohanec, M. (2012). A Method for Ranking Non-Linear Qualitative Decision Preferences using Copulas. *International Journal of Decision Support System Technology*, *4*(2), 42–58.

Moshkovich, H. M. & Mechitov, A. I. (2013). Verbal Decision Analysis: Foundations and Trends. *Advances in Decision Sciences*, *2013*, 9.

Nagel, S. S. (1993). *Computer-Aided Decision Analysis: Theory and Applications*. Westport: Praeger.

Omero, M., D'Ambrosio, L., Pesenti, R., & Ukovich, W. (2005). Multiple-Attribute Decision Support System Based on Fuzzy Logic for Performance Assessment. *European Journal of Operational Research*, *160*(3), 710–725.

Rajkovič, V., Bohanec, M., & Efstathiou, J. (1987). Ranking Multiple Options with DECMAK. In J. Hagwood & P. Humphreys (Eds.), *Efective Decision Support Systems* (pp. 49–60). Aldershot: Gower Technical Press.

Saaty, T. L. (2005). The Analytic Hierarchy and Analytic Network Processes for the Measurement of Intangible Criteria and for Decision-Making. In J. Figueira, S. Greco, & M. Ehrgott (Eds.), *Multiple Criteria Decision Analysis: State of the Art Surveys* (pp. 345–407). Springer Verlag.

Saaty, T. L. (2008). Decision Making with the Analytic Hierarchy Process. *International Journal of Services Sciences*, *1*(1), 83–98.

Saaty, T. L. & Vargas, L. G. (2012). *Models, Methods, Concepts & Applications of the Analytic Hierarchy Process* (2nd ed.) (F. S. Hillier, Ed.). International Series in Operations Research & Management Science. Springer.

Samuel, A. L. (1967). Some Studies in Machine Learning Using the Game of Checkers II: Recent progress. *IBM Journal of Research and Development*, *11*(6), 601–617.

Shachter, R. D. & Peot, M. A. (1992). Decision Making Using Probabilistic Inference Methods. In D. Dubois, M. P. Wellman, B. D'Ambrosio, & P. Smets (Eds.), *Eighth conference on Uncertainty in Artificial Intelligence*. Stanford, CA: Morgan Kaufmann Publishers Inc.

Skinner, D. C. (2009). *Introduction to Decision Analysis* (3rd ed.). Gainesville, Florida: Probabilistic Publishing.

Trdin, N. (2011). *Decision Support Model for Management of Water Sources* (BSc thesis, Faculty of Computer Science and Informatics, and Faculty of Mathematics and Physics, University of Ljubljana).

Trdin, N. & Bohanec, M. (2012). Extending the Multi-Criteria Decision Making Method DEX. In D. Petelin, A. Tavčar, & B. Kaluža (Eds.), *4th Jožef Stefan International Postgraduate School Students Conference* (pp. 182–187). Ljubljana, Slovenia.

Trdin, N. & Bohanec, M. (2013). Relational Multi-Attribute Models in DEX Methodology. In F. Ruiz (Ed.), *22nd International Conference on Multiple Criteria Decision Making* (p. 317). Málaga, Spain.

Trdin, N. & Bohanec, M. (2014a). New Generation Platform for Multi-Criteria Decision Making with Method DEX. In G. Phillips-Wren, S. Carlsson, F. Burstein, A. Respício, & P. Brézillon (Eds.), *17th conference for IFIP WG8.3 DSS*. DSS 2.0 – Supporting Decision Making with New Technologies. Paris, France: IOS Press.

Trdin, N. & Bohanec, M. (2014b). Numerical Relational Multi-Attribute Models in Qualitative Multi-Attribute Method DEX. In *20th Conference of the International Federation of Operational Research Societies* (p. 240). Barcelona, Spain.

Trdin, N. & Bohanec, M. (2015). nejctrdin / DEXx — Bitbucket. Retrieved from https://bitbucket.org/nejctrdin/dexx/overview.

Trdin, N., Bohanec, M., & Janža, M. (2013). Decision Support System for Management of Water Sources. In *Proceedings of the 16th International Conference Information Society IS 2013* (pp. 118–121). Ljubljana, Slovenia.

Wang, J. & Zionts, S. (2008). Negotiating Wisely: Considerations Based on MCDM/MAUT. *European Journal of Operational Research*, *188*(1), 191–205.

Xidonas, P., Askounis, D., & Psarras, J. (2009). Common Stock Portfolio Selection: A Multiple Criteria Decision Making Methodology and an Application to the Athens Stock Exchange. *Operational Research*, *9*(1), 55–79.

Yang, H.-L. (1995). Information/Knowledge Acquisition Methods for Decision Support Systems and Expert Systems. *Information Processing & Management*, *31*(1), 47–58.

Yang, J. B., Wang, Y. M., Xu, D. L., & Chin, K. S. (2006). The Evidential Reasoning Approach for MADA Under Both Probabilistic and Fuzzy Uncertainties. *European Journal of Operational Research*, *171*(1), 309–343.

Yu, L., Wang, S., Wen, F., & Lai, K. K. (2012). Genetic Algorithm-Based Multi-Criteria Project Portfolio Selection. *Annals of Operations Research*, *197*(1), 71–86.

Zadeh, L. A. (1965). Fuzzy Sets. *Information and Control*, *8*(3), 338–353.

Zupan, B., Bohanec, M., Demšar, J., & Bratko, I. (1999). Learning by Discovering Concept Hierarchies. *Artificial Intelligence*, *109*(1–2), 211–242.

Zupan, B., Bratko, I., Bohanec, M., & Demšar, J. (2001). Function Decomposition in Machine Learning. In G. Paliouras, V. Karkaletsis, & C. D. Spyropoulos (Eds.), *Machine Learning and Its Applications* (Vol. 2049, pp. 71–101). Berlin: Springer.

Žnidaršič, M., Bohanec, M., & Bratko, I. (2003). Categorization of Numerical Values for DEX Hierarchical Models. *Informatica*, *27*(4), 405–409.

Žnidaršič, M., Bohanec, M., & Zupan, B. (2006a). Higher-Order Uncertainty Approach to Revision of Probabilistic Qualitative Multi-Attribute Decision Models. In F. Adam, P. Brézillon, S. Carlsson, & P. Humphreys (Eds.), *IFIP WG8.3 International Conference on Creativity and Innovation in Decision Making and Decision Support* (pp. 817–833). London, UK: Decision Support Press.

Žnidaršič, M., Bohanec, M., & Zupan, B. (2006b). proDEX – A DSS Tool for Environmental Decision-Making. *Environmental Modelling & Software*, *21*(10), 1514–1516.

Žnidaršič, M., Bohanec, M., & Zupan, B. (2008). Modelling Impacts of Cropping Systems: Demands and Solutions for DEX Methodology. *European Journal of Operational Research*, *189*(3), 594–608.

Žnidaršič, M., Bohanec, M., Lavrač, N., & Cestnik, B. (2009). Project Self-Evaluation Methodology: The Healthreats Project Case Study. In M. Bohanec, M. Gams, V. Rajkovič, T. Urbančič, M. Bernik, D. Mladenić, ...A. Brodnik (Eds.), *Information Society — IS 2009*. Ljubljana, Slovenia: Institut Jožef Stefan.

Žnidaršič, M., Bohanec, M., & Trdin, N. (2012). Qualitative Assessment of Data-Mining Workflows. In A. Respício & F. Burstein (Eds.), *Fusing Decision Support Systems into the Fabric of the Context* (pp. 75–86). Frontiers in Artificial Intelligence and Applications. Anavyssos, Greece: IOS Press.

Žnidaršič, M. (2015). proDEX. Retrieved from http://kt.ijs.si/software/proDEX/Index.html.

Bibliography

Publications Related to the Thesis

Journal Article

Kontić, B., Bohanec, M., Kontić, D., Trdin, N., & Matko, M. (2016). Improving Appraisal of Sustainability of Energy Options — A View from Slovenia. *Energy Policy*, *90*, 154–171.

Conference Papers

Bohanec, M., Aprile, G., Constante, M., Foti, M., & Trdin, N. (2013). Decision Support Model for the Assessment of Bank Reputational Risk. In M. Gams, R. Piltaver, D. Mladenić, M. Grobelnik, F. Novak, B. Blažica, ... A. Brodnik (Eds.), *Information Society — IS 2013* (Vol. A, pp. 11–14). Ljubljana, Slovenia.

Bohanec, M., Aprile, G., Constante, M., Foti, M., & Trdin, N. (2014). A Hierarchical Multi-Attribute Model for Bank Reputational Risk Assessment. In G. Phillips-Wren, S. Carlsson, & A. Respício (Eds.), *17th conference for IFIP WG8.3 DSS* (pp. 92–103). Paris, France: IOS Press.

Bohanec, M. & Trdin, N. (2014). Qualitative Multi-Attribute Decision Method DEX: Theory and Practice. In *20th Conference of the International Federation of Operational Research Societies* (p. 239). Barcelona, Spain.

Bohanec, M., Trdin, N., & Kontić, B. (2015a). A Qualitative Multi-Criteria Model for the Evaluation of Electric Energy Production Technologies in Slovenia. In L. Zadnik Stirn, J. Žerovnik, M. Kljajić Borštnar, & S. Drobne (Eds.), *Proceedings of the 13th International Symposium on Operational Research SOR'15* (pp. 11–16). Bled, Slovenia: Slovenian Society Informatika, Section for Operational Research.

Bohanec, M., Trdin, N., & Kontić, B. (2015b). Strategic Assesment of Electric Energy Production Technologies in Slovenia Using Qualitative Multi-Attribute and Simulation Methods. In G. Pflug (Ed.), *OR2015 Vienna Conference: International Conference on Operations Research* (pp. 107–108). Vienna, Austria.

Kontić, B., Bohanec, M., Trdin, N., Kontić, D., Zagorc-Kontić, S., & Matko, M. (2014). Comparative Evaluation of Various Energy Options using Qualitative Multi-Attribute Models. In *20th Conference of the International Federation of Operational Research Societies* (p. 239). Barcelona, Spain.

Trdin, N. & Bohanec, M. (2012). Extending the Multi-Criteria Decision Making Method DEX. In D. Petelin, A. Tavčar, & B. Kaluža (Eds.), *4th Jožef Stefan International Postgraduate School Students Conference* (pp. 182–187). Ljubljana, Slovenia.

Trdin, N. & Bohanec, M. (2013). Relational Multi-Attribute Models in DEX Methodology. In F. Ruiz (Ed.), *22nd International Conference on Multiple Criteria Decision Making* (p. 317). Málaga, Spain.

Trdin, N. & Bohanec, M. (2014a). New Generation Platform for Multi-Criteria Decision Making with Method DEX. In G. Phillips-Wren, S. Carlsson, F. Burstein, A. Respício, & P. Brézillon (Eds.), *17th conference for IFIP WG8.3 DSS*. DSS 2.0 – Supporting Decision Making with New Technologies. Paris, France: IOS Press.

Trdin, N. & Bohanec, M. (2014b). Numerical Relational Multi-Attribute Models in Qualitative Multi-Attribute Method DEX. In *20th Conference of the International Federation of Operational Research Societies* (p. 240). Barcelona, Spain.

Trdin, N., Bohanec, M., & Janža, M. (2013). Decision Support System for Management of Water Sources. In *Proceedings of the 16th International Conference Information Society IS 2013* (pp. 118–121). Ljubljana, Slovenia.

Žnidaršič, M., Bohanec, M., & Trdin, N. (2012). Qualitative Assessment of Data-Mining Workflows. In A. Respício & F. Burstein (Eds.), *Fusing Decision Support Systems into the Fabric of the Context* (pp. 75–86). Frontiers in Artificial Intelligence and Applications. Anavyssos, Greece: IOS Press.

Reports

Kontić, B., Kontić, D., Zagorc, S., Matko, M., Dermol, U., Bohanec, M., & Trdin, N. (2014). *Ocena Vzdržnosti za Razvoj Energetike v Sloveniji do Leta 2030 s Poudarkom na Jedrski Tehnologiji. Book 1, IJS Report DP-11583*. Jožef Stefan Institute.

BSc Thesis

Trdin, N. (2011). *Decision Support Model for Management of Water Sources* (BSc thesis, Faculty of Computer Science and Informatics, and Faculty of Mathematics and Physics, University of Ljubljana).

Other Publications

Journal Articles

Grčar, M., Trdin, N., & Lavrač, N. (2012). A Methodology for Mining Document-Enriched Heterogeneous Information Networks. *The Computer Journal*, *56*(3).

Lavrač, N., Jesenovec, D., Trdin, N., & Mramor Kosta, N. (2008). Mining Spatio-Temporal Data of Traffic Accidents and Spatial Pattern Visualization. *Metodološki zvezki*, *5*(1), 45–63.

Pollak, S., Trdin, N., Vavpetič, A., & Tomaž, E. (2012b). NLP Web Services for Slovene and English: Morphosyntactic Tagging, Lemmatisation and Definition Extraction. *Informatica*, *36*(4), 441–449.

Conference Papers

Pollak, S., Trdin, N., Vavpetič, A., & Tomaž, E. (2012a). A Web Service Implementation of Linguistic Annotation for Slovene and English. In T. Erjavec & J. Žganec Gros (Eds.), *Information Society — IS 2012* (Vol. 100, pp. 157–162). Ljubljana, Slovenia.

Biography

Nejc Trdin was born on January 26, 1988 in Slovenj Gradec, Slovenia. He attended primary and secondary school in Velenje where he finished general gymnasium. During his primary and secondary education, he attended many national competitions in the areas of mathematics, physics, logic, English language and chemistry for which he won several bronze and silver awards. In 2007, he started his studies at the Faculty of Computer and Information Science, and Faculty of Mathematics and Physic, University of Ljubljana, Slovenia. He was enrolled in a 8-semester BSc program titled Interdisciplinary Studies of Computer Science and Mathematics. During his studies, he received four Dean's recognitions for exemplary study performance and another faculty award for exemplary finishing of studies. During his BSc studies he held a scholarship for talented students from the Jožef Stefan Institute. In this period, he was employed part-time by companies working on software projects for architectural design and design of web-pages, where he gained valuable experience in advanced computer technologies. He finished his studies and defended the BSc thesis in September 2011, titled Decision Support Model for Management of Water Sources, under the supervision of Prof. Dr. Andrej Bauer and co-supervision of Prof. Dr. Marko Bohanec.

In the Fall of 2011 he started his graduate studies at the Jožef Stefan International Postgraduate School, Ljubljana, Slovenia. He is enrolled in the PhD program entitled Information and Communication Technologies under the supervision of Prof. Dr. Marko Bohanec. During his PhD studies, he was a young researcher at the Department of Knowledge Technologies at the Jožef Stefan Institute, Ljubljana, Slovenia, and was involved in several European and national projects funded by the EU Commission and the Industrial sector. His research is in the field of multi-criteria decision modeling, in particular qualitative multi-criteria modeling method DEX. His current research — presented in this thesis — aims at developing a methodology for integrating possibly inaccurate or imprecise qualitative and quantitative knowledge, together with relationally connected alternatives, into a decision model described with DEX.