UHASSELT

KNOWLEDGE IN ACTION

Faculty of Business Economics

Master of Management

Master's thesis

Structured Literature review on DMN refactoring patterns

Victor Tatah Ntaryikeh

Thesis presented in fulfillment of the requirements for the degree of Master of Management, specialization Business Process Management

SUPERVISOR :

Prof. dr. Koenraad VANHOOF

UHASSELT

KNOWLEDGE IN ACTION

www.uhasselt.be
Universiteit Hasselt
Campus Hasselt:
Martelarenlaan 42 | 3500 Hasselt
Campus Diepenbeek:
Agoralaan Gebouw D | 3590 Diepenbeek

2019
2020

Faculty of Business Economics

Master of Management

Master's thesis

Structured Literature review on DMN refactoring patterns

Victor Tatah Ntaryikeh

Thesis presented in fulfillment of the requirements for the degree of Master of Management, specialization Business Process Management

SUPERVISOR :

Prof. dr. Koenraad VANHOOF

Disclaimer

This master thesis was written during the COVID-19 crisis in 2020. This global health crisis might have had an impact on the (writing) process, the research activities, and the research results that are at the basis of this thesis.

Abstract

With increasing interest and adoption of decision modelling Decision Model and Notation (DMN), a standard notation from Object Management Group (OMG) provide specifications in modelling both human and automated decisions so that organizational decision making can be readily depicted in diagrams (DMN models). More often, decision models developed manually or automatically did not adhere to expected standards. As such, these models need to be refactored for them to meet the defined style rules and convention standards. This paper proposes refactoring patterns that could be employed by decision modellers to restructure and improve DMN models. The proposed patterns enable modellers to effectively deal with complexity in specific modelling difficulties by making models better understandable and easier to maintain while preserving behaviour. The case of Hotel Het Menneke and an example of collateral eligibility were used to demonstrate the feasibility of the refactoring patterns.

Acknowledgement

I would love to express my sincere gratitude to my supervisor Professor Koen Vanhoof for his availability and overwhelming guidance and support towards the realization of this project.

My deepest appreciation equally to Miss Annelies Clijsters for her special attention and guidance throughout the difficult times during my study, as well as, my lecturers and course mates of the Faculty of Business Economics at Hasselt University for their valuable time in the course of my study.

Most especially, I am very grateful to the Almighty God for His enormous Grace that enabled me to go through my study.

Table of Contents

1 Introduction

Every organization has goals for which resources are placed for their attainment. To accomplish these goals, objectives and policies are set, activities, and procedures for their performance are laid down. The establishment and accomplishment of these goals, objectives, policies, activities, and procedures require decision-making. Decisions are made based on rules dependent on the environment in which the organization operates. The rules, originating from legislation or policies, eventually end up in many different places in the company (Slavova P.,2017). Individuals working in several points of action of the organization need to make decisions daily to accomplish their goals (R. Guizzardi et al.,2018).

Decision modelling expresses how a decision should be made as a rigorous, verifiable model. It formalizes decision-making so, it can be clearly and widely understood, managed, and used effectively. Decision modelling supports the documentation of an organization's decisions such that they can be made consistently, improved over time, and automated where appropriate (Taylor and Purchase., 2016). Documenting decisions is useful for capturing expert knowledge on decision-making to help understand how decisions have been made, possibly avoiding pitfalls, providing newly employed with the means to understand how the decisions have been made, so they can learn from them. And documenting the evolution of the organizational services and products in terms of the decisions that have been made in their regard (R. Guizzardi et al., 2018).

But often, decision models that are created manually, and even more so, those that are created automatically do not adhere to the defined standards. For instance, decision tables are much larger than allowed, sub decisions do not show a consistent level of granularity, or undesired hit policies are used. In such cases, DMN models must be refactored to meet the required style rules or conventions. Given the above setting, the main goal of this thesis is to identify refactoring patterns in DMN models.

This study is a structured literature review on DMN models with its objective of identifying refactoring patterns that could be incorporated by professionals in decision modelling to ensure a declarative, complete, and consistent DMN model. To achieve this goal, the main research question is "What refactoring patterns exist?" the following sub-questions are investigated:

- What is the required state of a decision in a DMN model?
- Are the qualities of DRD and related Decision Table dependent or independent?
- What style rules and/or convention standards required for a DMN model?

By examining DMN models against DMN standards, our contribution is threefold. Firstly, we synthesize and summarize the description of decision and required expected state of a DMN model, as well as, known style rules and convention standard adopted in DMN models. Secondly, we evaluate the dependence and independence of DRD and related Decision Tables(Boxed Expression). Thirdly, facilitate the identification of available DMN refactoring patterns.

This paper is structured as follows: Section 2 reviews related literature on decision modelling, refactoring, patterns, and defines key concepts. Section 3 presents the literature methodology employed in providing solutions to the research questions. Section 4 provides a literature review analysis. And finally, section 5 summarizes our contribution and recommendations for the future.

2 Literature Review

2.1 Decision and Decision Models.

2.1.1 Decision.

Decision-making is the process by which we form judgments and make choices to achieve our goals (S.A Jackson et al., 2016). The outcome of this process is a decision. Decisions are pervasive in business and vary widely in complexity and degree of business impact (Bruce Silver, 2016). All businesses rely on decisions in their operations (Blenko, Mankins, & Rogers, 2010) and could be classified into strategic, tactical, and operational.

Object Management Group (OMG) in Decision Model and Notation (DMN) version 1.2 (2019) defines a decision as;
The act of determining an output value (the chosen option) from several input values, using logic defining how the output is determined from the inputs.

Hence, a decision determines an output from several inputs by applying some decision logic (T. Debevoise & J. Taylor, 2014). In this paper, the term decision means an *operational business decision*. Operational business decisions are those that occur every day, typically affecting a single transaction or customer (Bruce Silver, 2016). It refers to a decision made repeatedly, during regular business activity, that controls or influences the organization's business behaviour (Taylor & Purchase, 2016). It is becoming increasingly apparent among practitioners that businesses should prioritize operational decisions (O. Hall, 2018). According to Bruce Silver (2016), DMN is intended for operational decisions, which cover a wide spectrum. For some, the answer is a single value – yes or no, a number, or a particular selected option. Note that, in DMN, a decision does not specify an action to be performed based on the result. It simply determines the value of the decision output, a data item.

2.1.2 Decision Models

According to Nagraj et al. (2007), decision models can be classified into two categories; (1) Deterministic models, which assume that all the relevant input data values are known with certainty. (2) Probabilistic models, which assume that some input values are not known with certainty. In this context, our focus is restricted to the former and limited to DMN. A decision model is not the same as a text-based requirement document. It communicates through structured diagrams and tables, not unstructured text.

A DMN model specifies the output value for any allowed combination of input values (Bruce Silver, 2016). The purpose of DMN as specified by DMN 1.2 is to model decisions so that organizational

decision-making can be readily depicted in diagrams, accurately by business analysts, and (optional) automated. And is used for modelling human decision-making, the requirement for automated decision-making, and for implementing automated decision making (OMG). OMG specifies that although there is a link between the process model and a decision model, it must be stressed that DMN is not dependent on the Business Process Model and Notation (BPMN). Its two-levels – decision requirement and decision logic – may be used independently or in conjunction to model a domain of decision making without reference to the business process.

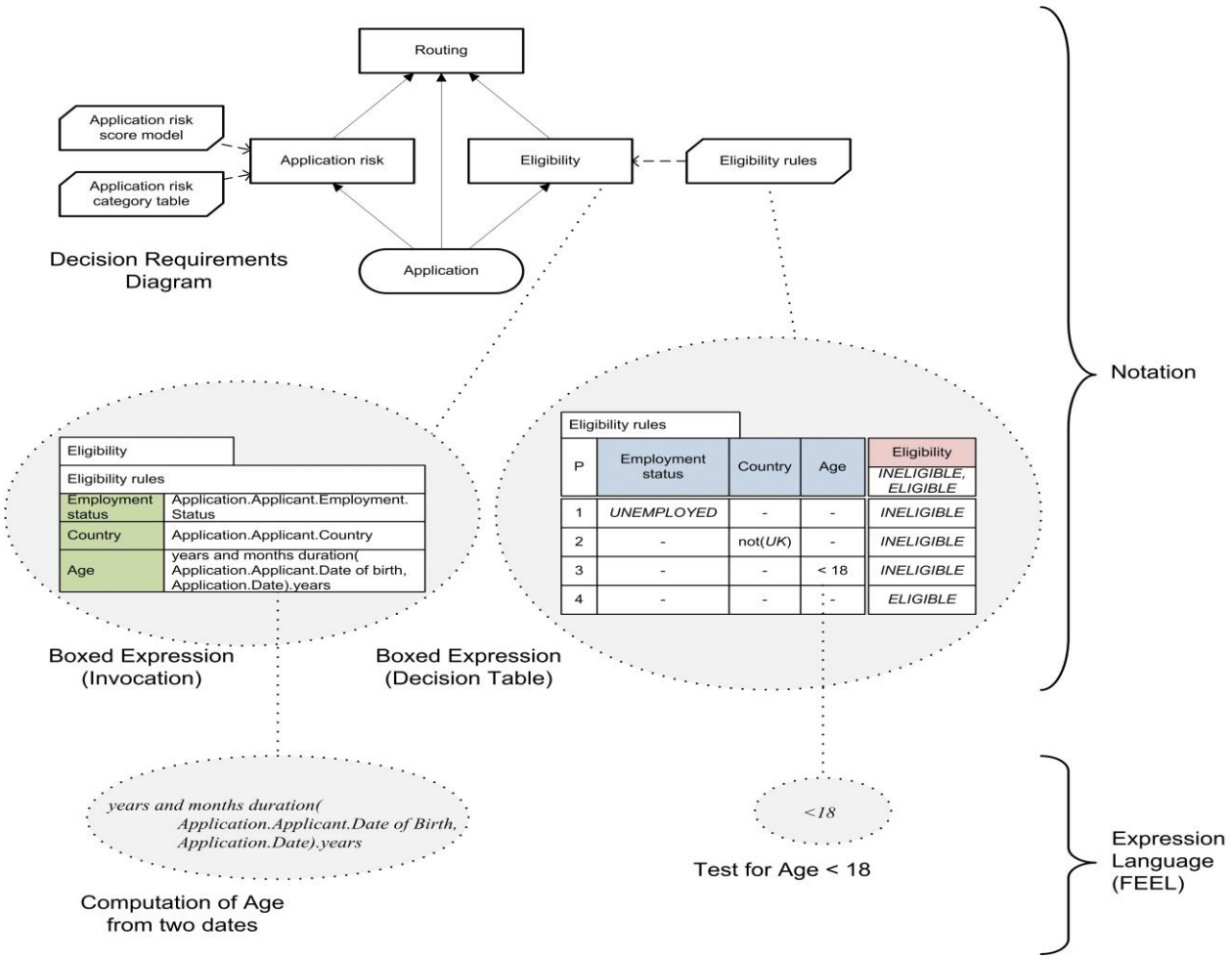

Figure 1. DMN Constructs.

Source: OMG

2.2. Decision Model and Notation(DMN).

Based on DMN version 1.2 (2019), DMN provides constructs spanning both decision requirement and decision logic modelling. For decision requirement modelling, it defines the concept of a Decision Requirement Graph (DRG) comprising a set of elements and their connection rules, and a corresponding notation: the Decision Requirement Diagram (DRD). For decision logic modelling, it provides a language called FEEL (Friend Enough Expression Language) for defining and assembling decision tables, calculations, if/then/else logic, simple data structures, and externally defined logic from Java and PMML into executable expressions with formally defined semantics. It also provides a notation for decision logic (boxed expression) allowing components of the decision logic level to be drawn graphically and associated with elements of a DRD. The relationship between these constructs is seen in figure 1.

A DRD is a combination of decisions, input data, knowledge sources, and business knowledge models (T. Debevoise & J. Taylor, 2014). It shows how a set of decisions depend on each other, on input data, and business knowledge models (BKMs). The use of business knowledge models (BKMs) to encapsulate decision logic is a matter of style and methodology, and decisions may be modelled with no associated business knowledge model (BKM) or with several. A BKM may contain any decision logic which is capable of being represented as a function. This allows the import of many existing decision modelling standards into DMN. Similar to BKMs, decision services may also be used to encapsulate decision logic for reuse inside the decision model (OMG). According to Bruce Silver (2016), DRD is a fabulous innovation. It can describe in a single diagram, the logic of "the business decision as a whole," in a purely declarative manner, independent of any BPMN model, even when it may be executed in multiple steps separated in time. By the business decision as a whole (end-to-end decision), Bruce means "the question this decision is supposed to answer," be it small or large.

The second level of DMN is decision logic. Decision logic is the means of selecting or calculating a specific outcome value in each scenario to which the decision is applied by using its inputs (Taylor & Purchase, 2016). It simply determines an outcome or set of output data values, for a given set of input data values (Bruce Silver, 2016). Decision logic may use one or more BKMs which encapsulate business know-how in the form of business rules, analytic models, or other formalisms. Using decision logic, business decisions, their interrelationships, the areas of business knowledge and data required by them, and the sources of the business knowledge can be specified in greater detail, to capture a complete set of business rules and calculations, and (if desired) to allow decision making to be fully automated (OMG).

One of the guiding principles of decision management has always been that decision logic should be declarative, meaning the order of evaluation does not change the outcome, as opposed to procedural,

specifying a particular order of evaluation (Bruce Silver, 2016). Declarative decision logic allows one to evaluate the decision outcome without regard to the order of evaluation. Bruce Silver (2016) notes that, technically, DMN does define two types of non-declarative decision tables, with a hit policy of either **F**irst or **R**ule order, meaning the first matching rule selects the output value. But the specification goes on to deprecate their use (because of this non-declarative behaviour). The details of how each decision's outcome is derived from its inputs must be modelled at the decision logic level. The decision logic level of a decision model in DMN consists of one or more value expressions. The elements of decision logic modelled as value expression include; tabular expressions such as decision tables, invocations, and literal (text) expressions (OMG). The tabular expressions or representations of decision logic are called boxed expressions, and they support several layouts and approaches including; Cross-tab, Rules as rows, and Rules as columns decision tables.

2.3. Decision Model Standard.

It is important to note that, Decision Models in this study are restricted to DRDs and Decision Tables (boxed expression) guided by but not limited to DMN standards. According to OMG, DMN is to provide a common notation that is readily understandable by all business users, create a standardized bridge for the gap between the business decision design and decision implementation, as well as, to ensure that decision model are interchangeable across organizations via an XML representation. Its authors brought forth expertise and experience from the existing decision modelling community and have sought to consolidate the common ideas from these divergent notations into single standard notation. The current standard of DMN is summarized in its conformance levels 1, 2, and 3 in DMN version 1.2 (2019).

Taylor and Purchase (2016) Clarify that OMG imposes some constraints on standards it adopts. One, in particular, is that the standard shall define only "What" and not "how" – it must avoid promoting a specific methodology for using the standard. Thus, it is capable of a standard to define the notation for a diagram but not to describe how one constructs such a diagram. In this context, according to Taylor and Purchase (2016), this means that DMN standard offers no specific tasks or steps that can be performed to build a Decision Requirement Model, no definition of what constitutes a "good" model beyond some basic structural constraints, and no suggestions as to how building a decision model might fit into a broader framework.

However, different conventions are being used by modellers in deriving "good" decision models per DMN standards and differences as regard presentation in what some professionals call style and methodology, method and style, best practices, advanced best practices, etc. Though there are some generally accepted decision modelling standards, Bruce Silver (2016) in his method and style illustrates that DMN standard falls short of the following: (1) Anything to do with methodology(although a bit of methodology sneaks in the back door with the Lending decision

example). (2) Anything to do with decision model testing, including consistency and completeness checking, test case generation, and simulation. R. Ghlala et al. (2017) argues that, DMN cannot deal with uncertainty and is limited to pre-defined decisions made from known criteria.

No matter the style, methodology, convention, or practice, there is a consensus that a "good" decision model should meet the standard of being complete, consistent, declarative, and represent the end-to-end decision logic. D. Calvanese et al. (2018) specifies two correctness criteria imposed on Decision Tables are that their rules should be disjoint and complete, meaning every possible input should match exactly one rule and a table violates these criteria if it has overlapping rules (multiple rules match a given input) or missing rules (no rule matches a given input). These are generally accepted with specifications as regards the "Hit" policy and DMN standard specify these. There is also a consensus on the following:

- A large number of inputs and outputs more than seven are is undesirable.
- A large number of rules more than a few hundred is cumbersome.
- Elements in decision models can be further emphasized. For example, setting its name in boldface, denoting the decision rectangle with a thicker line, making subject item lager or even use of colour fill. None of these conventions is part of the standard and should be minimized, consistently applied, and as simple as possible (Taylor and Purchase, 2016).
- Have a flow of dependencies progress in a consistent direction.
- Two or more specific rules (input entry cells) in a Decision Table can be merged in a single, more general rule.
- There is no concept of condition evaluation order in Decision Tables and no possibility of optimizing condition evaluation using order. Except in **F**irst tables (single hit) and **R**ule Order tables (multiple hit).
- If tables are allowed to contain overlapping rules, the table hit policy indicates how overlapping rules have to be handled and which is the resulting value (s) for the output name, to avoid inconsistency.
- When a style or convention is adopted, apply it consistently across all models.

2.4. Refactoring

Refactor or Refactoring is a term widely used by the software community. According to W.F. Opdyke (1992), refactoring is reorganization plans that support change at an intermediate level. They do not change the behaviour of a program. That is, if the program is called twice (before and after refactoring) with the same set of inputs, the resulting set of output values will be the same. Refactoring enables programmers to restructure a software system without altering its behaviour. Thus, it is typically used to improve code quality by removing duplication, improving readability, simplifying software design, or adding flexibility (B. Weber et al., 2011).

M. Fowler (2018) defines refactoring as the process of changing a software system in such a way that it does not alter the external behaviour of the code yet improves its internal structure. A. Christopoulon et al (2012) indicated that the refactoring process comprises a set of activities such as identification of the candidate code fragments and the refactoring (s), evaluation of behaviour preservation guarantees, application of the refactoring, and assessment of its effect on software quality. Refactoring techniques improves upon the internal quality of a model such that it becomes easier to read, maintain, but does not affect the model's semantics or external behaviour (B. Weber et al., 2011). In essence, when you refactor, you are improving the design of code after it has been written (M. Fowler, 2018). Redesign often affects the external quality and its results are visible to the customer. In contrast, refactoring techniques primarily impact internal quality and ensure conceptual integrity, and foster maintainability (B. Weber et al., 2011).

This study adopts the concept of refactoring from the software community (software engineering) to decision modelling. In this context, we define refactoring as:

The process of restructuring a decision model such that its internal quality is enhanced, ensures conceptual integrity, foster maintainability, without affecting the model's semantics and external behaviour.

A decision model here refers to DMN models. That is, a DRD and a Decision Table (Boxed Expression). This means refactoring is limited to restructuring the representation of a DMN model without any effect on the model's declarative, consistent, and completeness nature in representing the business decision as a whole. As such, refactoring neither resolve errors nor add functionality, but improves understandability and maintainability through behaviour preserving.

2.5. Patterns

Same as refactoring, a pattern is a term also widely adopted by the software community. The term was first adopted by the software community as a way of documenting recurring solutions to design problems (S. Demeyer et al., 2002). The concept of the pattern was developed by the architect Christoper Alexander for his particular domain (D.B. Roberts & R. Johnson, 1999). According to D.B. Roberts and R. Johnson (1999). Alexander says: "Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem in such a way that you can use this solution a million times over, without ever doing it the same way." As with Alexander's patterns, each design pattern entailed some forces to be resolved, and some trade-offs to consider when applying the pattern (S. Demeyer et al., 2002). S. Demeyer et al. (2002) defines a pattern as: "A motif, an event, or a structure that occurs over and over again." And indicated that, patterns turn out to be a compact way to communicate best practice: not just the actual techniques used by experts, but the motivation and rationale behind.

However, D. Riehle and Zullighoven (1996) holds a slightly different view of patterns, and defines a pattern as: "The abstraction from a concrete form which keeps recurring in specific non-arbitrary contexts." They indicated that their definition does not confine their notion of patterns to software design and is not restricted to a specific use of patterns. D. Riehle and Zullighoven (1996) argued that the definition of a pattern as " a solution to a recurring problem in a context" is geared toward solving problems in design. And to better understand their definition, they elaborated on what they meant by "form" and "context". The "form" of a pattern consisting of a finite number of visible and distinguishable components and their relationships. Defining the notion of "form" through its representation as a set of interacting components and their relationships, a component need not be a software component but can be any kind of technique or non-technical entity.

Indicating that a pattern is a form that appears in a context, D. Riehle and Zullighoven (1996) specified that, pattern and context are complementary: context constraints the pattern and gives birth to its form while the form fits only into certain contexts. Changing the context will change the form and changing the form has to go hand in hand with changing the context. Patterns should be integrated into a given context, that is, assumptions about the environment which embeds the pattern are highly relevant (V.A. Aalst et al., 2003). Patterns are in no way invented, they are discovered or "mined" from and existing system (M.O. Cinneide, 2001).

Taylor and Purchase (2016) hold the view that there are some demanding requirement structures and decision modelling challenges that reoccur across many business domains. These reoccurring problems can be addressed by the application of specific techniques and reusable model structures or templates. To Taylor and Purchase (2016), the most frequently used model templates are often referred to as "patterns." Likewise, the DMN Method by Bruce Silver (2016) recommended standardizing on a set of common decision logic patterns that both modellers and consumers of the decision models will learn to recognize. According to Taylor and Purchase (2016), documented patterns can be shared with less experienced modellers and their application to a specific problem can be debated without ambiguity, they can be refined over time, and their use can be combined with other patterns. Some patterns are very specific to certain problem contexts and others are widespread and almost commonplace. And using patterns, business analysts can cooperate and evaluate rival models more effectively.

Once you learn how to model these patterns, breaking even complicated decision logic becomes far easier (Bruce Silver,2016). Thus, this study adopts the definition of a pattern from the above concepts as follows:

A pattern refers to a specific technique and reusable model structure or template that can be employed to address decision modelling challenges that occur across many business domains.

The techniques and reusable model structures or templates here are limited to decision models using DMN specification as required by DMN version 1.2 prescribed by OMG. And focused on addressing challenges involved in developing and restructuring DMN models.

2.6. Refactoring Patterns.

Based on our above considerations and specifications concerning refactoring and patterns, we refer to refactoring patterns as:

Those techniques and reusable model structures or templates that can be used to restructure the representation of a DMN model without any effect on the model's semantics or external behaviour.

The refactoring pattern used should enhance the understanding of the decision model and foster maintainability through behaviour preservation. According to Taylor and Purchase (2016), the following refactoring patterns could be used based on the specific decision modelling challenge:

(i) **Divide and Conquer Pattern.** This pattern splits Decision Tables (or decisions) into subordinates of much small and more manageable size using a partition – a criterion for performing a split that preserves behaviour. The best split divides the decision into sub tables with different expertise jurisdictions that can evolve independently. The four variants of this pattern are specialization, tall table, non-cohesion, and order complex. It is a means of splitting a single, over-complicated decision that violates complexity best practices into several sub-decisions that together perform the same task.

(ii) **Exception Based Logic Pattern.** This pattern is required where there exist special cases. Special cases, even if rare, it can account for much of the work of a decision modeller. The chief symptoms of this are that the structure of the Decision Requirement Models (and often the associated Decision Logic Models) is dominated by special cases that only occur in a tiny fraction of the presented data and which keeps changing. This pattern requires that the decision model be dominated by decisions (and their logic) of the mainstream cases and that exceptional case be specified and handled "separately" with a purpose-built, localized decision, and then reintegrated with the overall outcome. It is a variation of Divide and Conquer that takes into account the frequency of use of special cases and is DRD rather than Decision Table oriented.

(iii) **Positive Validator Pattern.** This pattern is used to validate data in a compact way that separates this concern from the rest of the decision model and act as documentation for data quality constraints in a way that traditional (negative) validation logic cannot. It assumes that all attributes of input data are individually available in a DMN context and that context variable attributes contain a list of their names. Using these context variables, allows DMN text rules to replace Decision Tables in which many columns would be needed if each were dedicated to specific attributes. Positive validator ensures that validation

conditions are both compact and expressed as assertions of the data quality requirements, rather than negatives.

In the same light, Bruce Silver (2016) recommended a pattern the same as Divide and Conquer and he called this pattern: Action sub table Pattern.

(iv) Action Sub table Pattern. An action sub table occurs when you have a Decision Table in which an output entry references a supporting decision. If the decision rule containing that output entry does not match, there is no need to evaluate the supporting decision. But if it does match, then you must execute the supporting decision – the action table –to determine the output entry value. The supporting decision is called an action sub table because its execution is effectively triggered by a rule in the parent-level decision in the DRD, a form of backward chaining. While some people find them confusing, action sub tables provide an elegant way to reduce the size of a Decision Table.

2.7. Refactoring Determinants.

The benefits of refactoring or the use of a refactoring pattern cannot be overemphasized. It enables the removal of duplication, improve readability, and enhance the maintainability of decision models. And maintenance will become increasingly difficult over time if no techniques for quality improvement are provided (B. Weber et al., 2011) for DMN models. The following factors influence refactoring and/or the adoption of a particular refactoring pattern:

According to Taylor and Purchase (2016), refactoring is called for when Decision Tables are not good because they:

- Obscure individual decisions by mixing them with others.
- Discourage the independent evolution of the decision.
- Make reuse of individual decisions harder.
- Make validation, testing, and maintenance of individual decisions harder.
- Repetition of conditions and bloat. The repetition suggests an inherent policy that should be made explicit.

S. Demeyer et al (2002) also stated that a refactoring pattern could be adopted in order:

- To exploit new technology, such as emerging standards or libraries, as a step toward cutting maintenance costs.
- To reduce human dependencies by documenting knowledge about the system and making it easier to maintain.

Moreover, refactoring could be employed because of challenges driven by the release of new regulation, policy, or feeder system.

3 Literature Review Methodology.

The literature review provides a historical perspective of a research domain and a major contribution to research advancement. Develop a targeted strategy to find key publications quickly and efficiently and then expand from there, prioritizing and keeping records as you go (H. Newing, 2010). This study explored related theoretical literature as regard decision modelling in view of identifying decision refactoring patterns that decision modellers could incorporate to restructure and improve DMN models. To achieve the goal of this research, four research questions were formulated.

The first research question, what decision refactoring patterns exist? This entailed identifying different existing refactoring patterns that decision modellers could use based on the specific modelling challenges they face, as a technique of restructuring a DMN model to improve understanding of the model. While preserving its conceptual integrity in its peculiar context, as well as, preserving its semantics and external behaviour. This was the main research question of this study and which guided the other research questions.

What is the required state of a decision in a DMN model? Was the second research question. An understanding of a decision as described by a DMN model (DRD and Decision Table), was the major concern of this research question. Its purpose was geared towards not only understanding the attributes of a DMN model, but most importantly the expected state (Declarative, Consistent, Complete, and End-to-End) of a decision as represented in a DMN model.

The third research question was, Are the qualities of DRD and related Decision Tables (Boxed Expression) dependent or independent? This research question involved an understanding of the connection between the DRD and related Decision Tables (boxed expression) to describe their possible dependence and independence in modelling a decision. The goal was to identify different possibilities that DRD and related Decision tables could be dependent or independent as regard to decision modelling.

What style rules and/or convention standards are required for a DMN model? This was the fourth research question. Based on the required elements of a DMN model, this research question involved understanding the attributes of a DMN model and how they could be structured with respect to DMN specification and best practices. The goal was towards identifying acceptable style rules, practices, and specifications in representing elements in a DMN model that enable distinction and facilitate reading and understanding while avoiding complexity.

The above research questions formulated in relation to achieving the goal of this study, guided the search for comprehensive literature in the context of decision modelling as regards DMN models. This step corroborated with the following criteria in reviewing related literature recommended by P. Cronin

et al (2008): (1) formulate the research question. (2) set inclusion or exclusion criteria. (3) select and access the literature. (4) assess the quality of the literature. (5) analyse, synthesize, and disseminate the findings. This approach is similar to the recommended criteria by Y. Levy and T.J. Ellis (2006). Moreover, the existing literature is an important element of all research (A. Bryman, 2016).

In searching for comprehensive literature, key words considered to likely capture related articles to this study were used. These guided the focus and facilitated the identification and retrieval of relevant material as per this study. The key words included; decision modelling, decision models, decision tables, decision patterns, decision structured tables, decision refactoring, decision refactoring patterns, refactoring patterns. This is in line with "concepts represent key areas around which data are collected" (A. Bryman, 2016). The methodology adopted in reviewing related literature in this study involved; (1) Identify information search sources. (2) Search for related literature. (3) Analyse, assess, and sort articles. (4) Extract and summarize information or data. (5) Synthesize and write the review.

The search for studies in written expression was part of the literature search. Open access and electronic databases such as Google Scholar, Research Gate, Science Direct, and ProQuest were searched. According to R.J. Chenail (2011), complement your electronic searches with systematic reviews of the references cited in the articles collected to locate additional sources. This was done in accordance with Webster and Watson (2002) structured approach to determine source material for review; (1) Leading journals. (2) Go backward (by reviewing citations of identified articles). (3) Go forward(by using the web of science to identify articles citing the key articles identified in the previous step).

Over 70 articles were identified. With this number, sorting was imperative. Analysis, assessment, and sorting of these articles were done by actual reading. And in accordance with the formulated research questions, articles deemed appropriate were retained. This helped reduce the number reviewed to be manageable. Examining the sorted articles to assure the quality by a thorough reading of their content, the emphasis was on literature related to DMN specification, refactoring, and patterns. Extracted data from chosen articles were structured, aggregated, and synthesize to enhance coherence in literature writing.

4. Literature Review Analysis.

4.1 Decisions and required state of DMN models.

The term decision as earlier stated in this paper refers to operational business decisions. That is, business decisions that occur every day or are repeatedly, typically affecting a single transaction or during regular business activities that influence the organization's business. Also, our adopted definition of decision as:
The act of determining an out output value (the chosen option) from several input values, using logic defining how the output is determined from the inputs (OMG).

This definition is complemented by P. Slavova (2017) by defining a decision as a specific type of behaviour element. That is a process or function that is governed by a set of decision rules. In DMN, a decision simply determines the value of the decision output and does not specify any action to be performed based on the output. J. Taylor and J. Purchase (2016) clarifies this by stating the aims of DMN to be: to express business logic precisely, in sufficient detail to support verification and execution, while still retaining ease of comprehension for business subject matter experts by excluding technical implementation details.

According to T. Debevoise and J. Taylor (2014), the structure of a decision and the logic can be modelled with DMN. But R. Ghlala et al. (2017) holds that DMN cannot deal with uncertainty and is limited to predefined decisions made from known criteria. And R. Guizzardi et al. (2018) argues that modelling decisions with DMN and using a modelling language to document decisions may prove useful to allow a more consistent, structured, and uniform view about decisions taken within an organization.

Von Halle and Goldberg (2009) in The Decision Model (TDM) specified that a decision model by definition represents all of the business logic(decision logic) or business rules behind only one business decision. Decision making is understood as a sub-area of problem-solving, which requires the selection of a single one of at least two alternatives, that is, mutually exclusive actions, which are designed to solve a problem by achieving one or more goals (Haendler & Frysak, 2018). DMN decouples decisions and control flow logic, and it opens room for dynamic management of decisions (Figl et al., 2018). As well as reduces complexity and provides a decision model that is more precise and clear (Bossuyt & Gailly, 2017: Figl et al., 2018).

In modelling decisions, a DMN model is required to be declarative, consistent, complete, and should represent the end-to-end decision logic. According to J. Taylor and J. Purchase (2016), DMN is a declarative approach to modelling decision making. Declarative means the order of evaluation (evaluating the decision outcome) does not change the outcome, as opposed to procedural, specifying a particular order of evaluation (Bruce Silver, 2016). A declarative solution only specifies what needs

to be done, with no details as to how it is to be carried out (done) because the sequence is irrelevant to arriving at the correct result (von Halle & Goldberg, 2010).

More so, OMG specified that, regardless of how expected input values are modelled (in decision tables), input values should be exclusive and complete. Exclusive means that input values are disjoint. Complete means that all relevant input values from the domain are present. And Y. Tang (2009) stated that a Decision Table is complete when all the possible combinations of conditions are listed as decision columns. Also, a well-designed Decision Table needs to be consistent. That is syntactical or structural consistency (Y. Tang, 2009). Syntactical consistency refers to a Decision Table without overlapping rules, rule gap, cyclical rules, invalid attribute values, and unreachable conditions/decision rules. By end-to-end decision logic, Bruce Silver (2016) simply means "the question the decision is supposed to answer" be it large or small.

4.2 DMN and BPMN Related.

That growing interest in business process modelling and decision modelling have led to enormous research contributions that have enabled the development of BPMN and DMN, which have become valuable considerations in managing organizations from an operational perspective. According to OMG, DMN creates a standardized bridge for the gap between the business decision design and decision implementation, and DMN notation is designed to be usable alongside the standard BPMN business process notation. BPMN's scope comprises the business logic containing information on what activities need to be executed in which order, by which resources, utilizing which data objects. While DMN covers the decision logic modelling by specifying which decision is taken based on which information, where to find it, and how to process this information to derive the decision result (K. Batoulis et al., 2015).

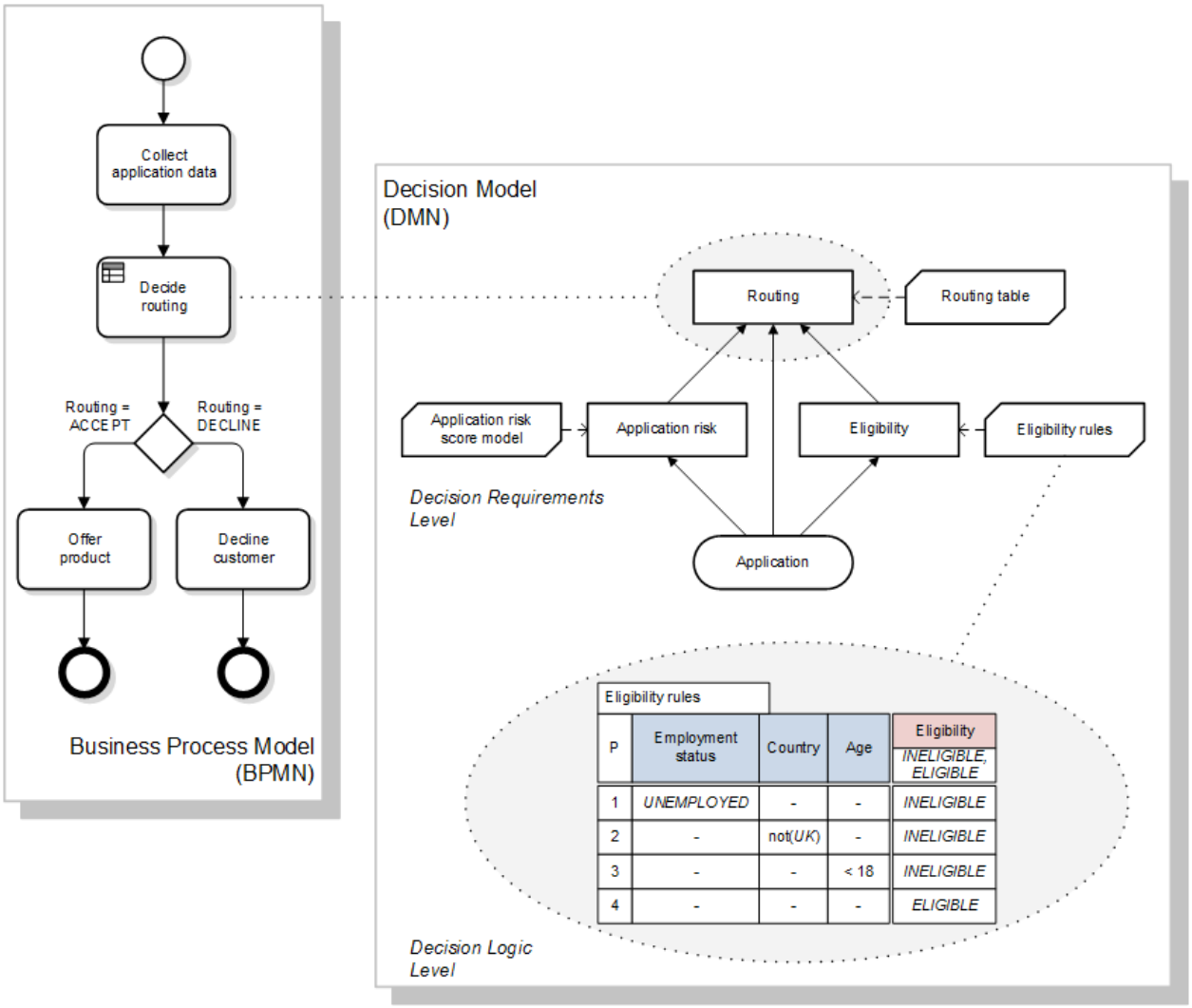

Figure 2. Aspects of Modelling. Source: OMG

Figure 2 shows the linkage between BPMN and DMN (Decision Requirement Level and Decision Logic Level) as regard aspects of modelling to explain their relationships. According to K. Figl et al (2018), DMN is a standard for representing operational decision of day-to-day business operations and complements BPMN with a notation for modelling decision logic and dependencies between decisions and data elements. With these, the deliberate separation of business process and business decisions into their models proved crucial to the ability to manage business logic and also transformed the business process into one that is decision-aware (von Halle & Goldberg, 2010). As such, by separating decisions from the process control-flow, the decision logic is specified in a declarative way. Also, the decision rules can be explicitly specified. This means that the decision logic can be reused, be adjusted to evolving within an ever-changing environment and be used as justification for the choices being made (S. Mertens et al., 2015).

In a similar note, F. Hasic' et al (2020) holds that DMN aims at providing a clear and simple representation of decision in a declarative form and offers no decision resolution mechanism of its own. Rather, the invoking context, for example, a business process is responsible for ensuring a correct invocation and enactment of the decision. As well as, ensuring data processing and the storage and propagation of data and decision outcomes throughout the process.

4.2.1. DRD and related Decision Table (boxed expression) dependence.

According to M. Deryck et al (2018) and F. Hasic' et al (2020), DMN consist of two levels that can be used in conjunction. First, the decision requirements level represented by the DRD which depicts the requirements of decisions and the dependencies between elements involved in the decision model. Second, the decision logic level which presents ways to specify the underlying decision logic. Decision requirements level presents how decisions depend on each other, and what input data is available for the decisions. And decision logic level describes the actual decision logic applied to take the decision (K. Batoulis et al., 2015).

More so, DRD represents the relationship between decisions through their information requirements and defines decision requirement through constructs of decisions, business knowledge models (BKMs), knowledge sources, input data, information requirement, knowledge requirement, and authority (K. Figl et al., 2018 and DMN version 1.2, 2019). The decision logic level represents the logic of a single decision in the form of a boxed expression. One of the most widely used representations for decision logic is a Decision Table (K. Figl et al., 2018).

With respect to the views of authors (including members of the submission team) as indicated by OMG, DMN specification acknowledges that decision making has an internal structure that is not conveniently captured in either of BPMN and decision logic perspectives. And DRD is to form a bridge between business process models and decision logic models. OMG further stressed that DMN is not dependent on BPMN, and its two levels may be used independently or in conjunction to model a domain of decision making without any reference to business processes.

Bruce Silver (2016) argues that the DMN specification makes the linkage between the DRD and BPMN explicit, but believes, it distorts their true relationship. That, the DRD is intended to create "a bridge between business process models and decision logic models," but the distortion comes from the fact that the primary purpose of DRD is not to suggest the procedural aspects of executing the end-to-end decision logic. Its primary purpose is to describe the decision logic of the business decision as a whole. The DRD structure may be influenced by the steps in its execution, but the DRD can describe the end-to-end decision logic on its own. Bruce Silver (2016) went further to state that, in a business decision evaluated as a single stateless action, that is, implemented as a single decision service, there is no distortion. The problem arises in business decisions evaluated in multiple steps separated in time,

corresponding to multiple tasks in the BPMN which could occur for many reasons (the DRD includes human decision, the DRD includes external decisions).

Furthermore, OMG clarifies that for human decisions, decision making can be broken down into a network of interdependent constituent decisions, and modelled using DRD. The decision in the DRD would probably be described at quite a high level using natural language rather than decision logic. But in some cases, it may be possible to define specific rules or algorithms for decision making. These may be modelled using decision logic (for example, business rules or decision tables) to specify BKMs in the DRD, either descriptively (to record how decisions are currently made or how they were made during a particular period of observation) or prescriptively (to define how decisions should be made or will be made in the future).

Also, Bruce Silver (2016) indicated that Lary Goldberg argued that once a DRD contains over thirty decisions, it becomes too difficult for stakeholders to understand, much less to rely on for maintaining the logic. Besides, there is considerable value in organizing complex decision logic as a hierarchy of supporting decisions. As opposed to a single giant Decision Table in which all of the input expressions reference only input data. It will be immense and essentially unmanageable (nearly impossible to analyse, test or modify). The DRD plus the boxed expression form a complete, mostly graphical language that completely specifies decision models (OMG).

4.2.2 DRD and related Decision Table (boxed expression) independence.

As regards the independence of a Decision Table to a DRD, OMG clarifies that the use of DMN for modelling the requirements for automated decision making is entirely prescriptive, and there is more emphasis on the detailed decision logic. The decision logic must be complete. That is, capable of providing a decision result for any possible set of values of the input data for full automation. However, for partial automation, where some decision making remains the preserve of personnel, interactions between human and automated decision making may be modelled using collaborations (DRD, Decision Table, and BPMN). More so, decision requirements level is often sufficient for business analysis of a domain of decision making, to identify the business decisions involved, their interrelationships, the areas of business knowledge and data required by them, and the sources of the business knowledge. To capture a complete set of business rules and calculations, the same components may be specified in greater detail using decision logic, and (if desired) to allow the decision making to be fully automated (OMG).

J. Vanthienen and G. Wets (1994) argued that there is no need to restrict the use of Decision Tables to the transformation from an expert system to Decision Tables, to obtain smooth validation or execution efficiency. It is superior to model knowledge utilizing Decision Tables from start, to validate and optimize the tables and then to implement the system. For example, using an expert system shell.

Decision Tables are considered the core concept of DMN, and they contain the necessary information to automate decision making. The DRD is mainly used to get a high-level understanding of the structure of the problem domain but does not contain additional information (M. Deryck et al., 2018).

4.3 Style rules and convention standards.

Several tools are available for modelling, execution, and analysing classical decision models. Some of which include; Signavios's DMN editor, OpenRules, Trisotech, Camunda, ProLoga, etc. With our limitation to DMN models, we referenced the DMN standard (DMN version 1.2, 2019) and some renowned contributors to decision modelling. In this context, some methodology and stylistic guidelines are essential to the effective use of DMN.

Bruce Silver (2016) holds the view that OMG standards normally try to avoid anything that smacks of *methodology* or *stylistic recommendations*. According to him, the thinking is that tool vendors and practitioners in the space simply have too much investment in their *method and style* to allow OMG to declare elements of that practice invalid unless they violate the metamodel or semantics of the standard. J. Taylor and J. Purchase (2016) also indicated that OMG imposes some constraints on standards it adopts. One, in particular, is that the standard shall define only *what* and not *how*; it must avoid promoting a specific methodology for using the standard. J. Taylor and J. Purchase (2016) further clarifies that the tendency to believe that the only reason for building a decision model is to capture business rules or decision logic is misguided. That is, experience with decision modelling has shown that decision models have multiple use cases, and even if the decision logic is not specified for every decision in the model, the model itself still has value. Bruce Silver (2016) suggests that there is considerable value in organizing complex decision logic as a hierarchy of supporting decisions. J. Taylor and J. Purchase (2016) complement this by recommending the necessity to avoid repetition of individual components that connect with many others, and have a *flow* of dependency progress in a consistent direction (top-down, bottom-top, left to right, insight out) rather than be haphazard or circuitous. To them, the layout of a diagram should always be chosen to maximize ease of comprehension without overloading it with meaning. If a diagram is focused on a specific subject decision, then this subject should either be at the top or, the middle of a diagram to emphasize its importance.

OMG clarifies that, in displaying elements in the Decision Table and depending on its size, a Decision Table could be presented horizontally (rules as rows), vertically (rules as columns), or crosstab (rules composed from two input dimensions). Crosstab can only have the default hit policy. Regardless of how input values are modelled, they should be exclusive (disjoint) and complete. The sequence of the rules in a Decision Table does not influence the meaning, except in **F**irst tables (single hit) and **R**ule order tables (multiple hit). J. Taylor and Purchase (2016) indicated that the difference between the

decision tables (rules as rows or rules as columns) is purely a matter of format. And it is better to adopt a single format for use.

To avoid name collisions and ambiguity, OMG recommend the name of a variable (input variable) must be unique within its scope, the name is the only visual link defined between DRD elements and boxed expressions. Boxed expressions are defined recursively. That is, they can contain other boxed expressions. In addition, Bruce Silver (2016) recommend a naming convention in which the data type of the variable is suggested by the variable's name. To OMG, line style is normative. There is a double line between the input clauses and the output clauses, continuing between the input entries and the output entries. J. Taylor and J. Purchase (2016) complement this by stating that, a double horizontal line separate the header row from the rules, double vertical line separate the conclusion column (s) from the condition (s). OMG adds that colour is suggested but does not influence the meaning. It is considered good practice to use different colours for input clauses, the output clauses, and the annotation clauses, and another (or no) colour for input, output, and annotation entries. Rule numbering is required for a table with hit indicated F (**F**irst) or R (**R**ule order), because the meaning depends on rule sequence and crosstab tables require no rule numbers.

Bruce Silver (2016) further recommends the use of compound decision tables whenever the values of multiple outputs are simultaneously determined by a single combination of input values. This is supported by OMG specifying that Decision Tables with compound outputs support only the following hit policies; **U**nique, **A**ny, **P**riority, **First**, **O**utput order, **R**ule order, and **C**ollect without operator. Furthermore, a dash symbol ("-") could be used to mean any input value. That is, the input is irrelevant for the containing rule. Adjacent input cells from different rules, with the same content and same (or no) prior cells can be merged. Rule output cells cannot be merged (except crosstab). These are supported by J. Taylor and J. Purchase (2016) and they went further to clarify that, the use of the ("-") marker in a conclusion cell has a different meaning to its use in a condition. And accepted that the use of ("-") in the conclusion cell is not technically supported by DMN standard at their time of writing.

According to Y. Tang and R. Meersman (2008), ambiguity, inconsistency, and misunderstanding arise when a Decision Table gets too large. Bruce Silver (2016) concur with this while citing his argument with Lary Goldberg, that a DRD with over thirty decision nodes become unmanageable and end-to-end decisions larger than that should not attempt to define their logic in a single DRD. J. Taylor and Purchase (2016) adds that a large number of inputs and outputs more than seven in a Decision Table is usually undesirable. A larger number of rules more than a few hundred is cumbersome.

Further required standards by OMG include; The expression language of a decision logic level may, but need not, be formal or executable. The use of BKMs to encapsulate decision is a matter of style and methodology and decisions may be modelled with no associated BKMs, or with several. Decisions may

require multiple BKMs and a BKM may require multiple other BKMs. Similar to BKMs, decision services may be used to encapsulate decision logic for reuse inside the decision model.

Other styles and convention recommendations compared to the DMN standard could be seen in Table 1.

	DMN	Vanthienen	Ross	TDM
Condition heading	Input expression, possibly involving multiple variables, or comparison test	Variable name or comparison test	Phrased as a question	Variable name
Conclusion heading	Decision name (output variable name)		Phrased as a question	Variable name
Layout	Choice of Rules-as-rows, Rules-as-columns, or Crosstab	Rules-as-columns preferred	Crosstab preferred except for complex tables	Rules-as-rows only
Explicit scope, restrictions, exceptions	No	Not recommended in general; occasionally useful.	Yes	Yes
Incomplete tables	Allowed but not encouraged. Default output optional.	No. Default output sometimes useful but confusing to business users.	OK if default output defined.	No, unless out of scope
Hit policy *Unique*	Default, but not explicitly preferred	Strongly preferred	Preferred	
Hit policy *Any*	Yes	No, "because redundancy is present (not well-normalized)."	No	Yes if hits are in separate rule patterns
Hit policy *First* , *Priority*	*First* allowed, *Priority* is preferred	*First* not allowed; *Priority* discouraged but better.	No	No
Compound output	Yes	Yes	Not recommended	No
Maximize contraction	Not required	Yes, not only for pairs of rules but blocks of rules	Yes	Yes
Optimize cell merging, ordering for visual scanning	Not required	Yes	Not required	No
Variable definitions based on glossary	No	No	Yes	Yes
Unique concern	Executable but business-friendly	Ease of visual scanning for completeness, consistency	Single-sourcing, structured vocabulary	Normalization (remove implicit dependencies)

Table 1. DMN vs recommendations/preferences of decision table experts
Source: Bruce Silver (2016)

4.4 DMN refactoring patterns.

According to A. Riehle and H. Zullighoven (1996), patterns and contexts are complementary. S. Demeyer et al (2002) argued that, as with Alexander's patterns, each design pattern entailed some trade-offs to consider when applying the pattern. Below are some of the refactoring patterns that could be applied to restructure DMN models.

4.4.1 The case of Hotel Het Menneke.

Hotel Het Menneke is a 3-star hotel located at the city centre of Hasselt in Belgium, in a renovated 1900 mansion just a minute walk from the Grote Markt and St Quintinus Cathedral and a 10-minute walk from the rail station. The nearest bus stop is an approximately 4-minute walk. The hotel offers accommodation with free high-speed Wi-Fi and has five rooms. 2 rooms of 16 square meters, 2 rooms of 25 square meters, and 1 room of 40 square meters. All rooms are equipped with air-conditioning, a mini-fridge, a flat-screen satellite TV, hairdryer, private bathroom, coffee/tea maker, and a closet. Extra amenities such as shower gel, shampoo, body lotion, and the city map are provided to enhance the comfort of clients.

Room service is available from 11 am to 2 pm 24/7. Breakfast is served at the on-site restaurant and payment is included in the price of the room. For launch and supper, clients are advised on their choice of nearby restaurants. The hotel is ideal for business owners coming from nearby cities who wish to shop for resale. For tourists, all main tourist attractions and opportunities are within walking distances and a few minute drive. Reservation of room is done online through the website of the hotel or commercial websites including; Booking.com, Master, Expedia.be Maxmind.com and Portal Toerisme Limburg.

Customers are required to provide their contact information, choose their room preferences, and the system will assemble some offerings in line with the customer's liking. From the list of available offers, the customer makes a selection and conduct payment. Every client is expected to effect payment of any reserved room one week before the arrival date. If not, the reservation is cancelled and the room is available for the next available client. Once payment is effected, the room will be marked as booked.

It is very important to note that, our main focus, in this case, is on the decision models and not the process models. Reasons while our process models are of a very low level.

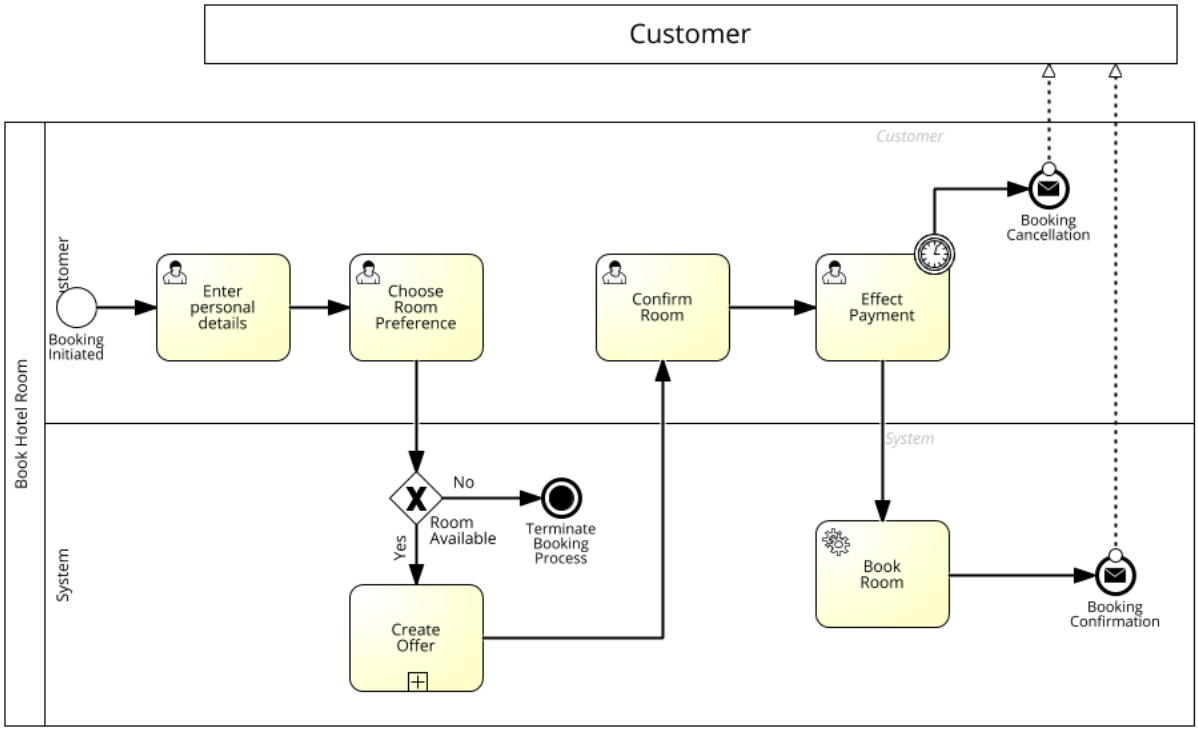

Figure 3. Process: Book Hotel Room.

In the process of booking a hotel room, after the client must have chosen the preferred room, the system creates an offer based on the chosen and the client is expected to confirm this offer and effect payment, after which the room is booked for that client. Figure 4 shows the process of creating an offer for a client.

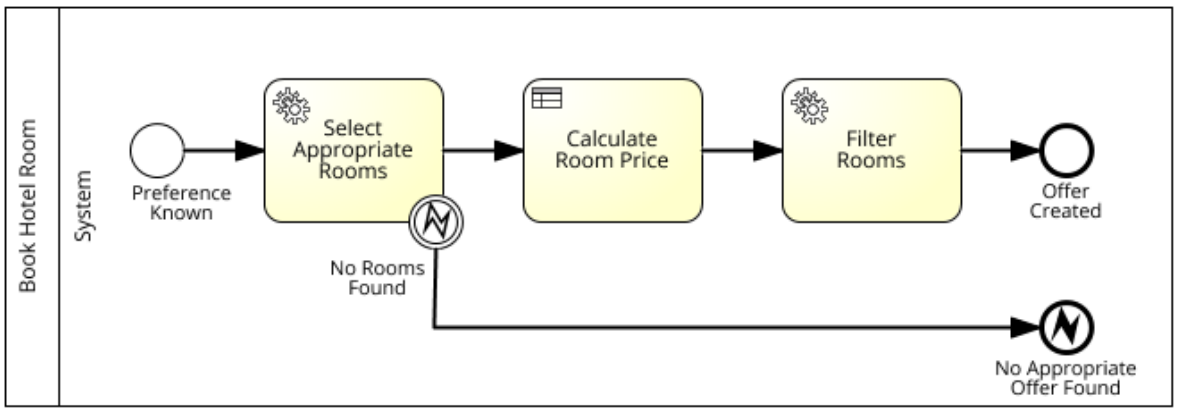

Figure 4. Subprocess: Create Offer.

Within the subprocess create an offer, the system will calculate the price of each room in the selection and retain only those rooms that satisfy the possible price ranges from the preferences of the client makes the offer. The client is then expected to confirm the preferred room and effect payment.

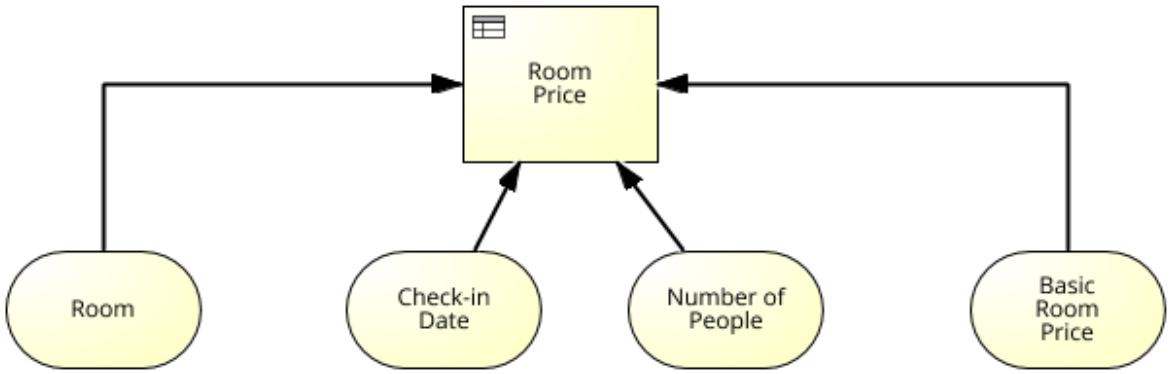

Figure 5. DRD. Decision: Calculate Room Price.

Figure 5 is the DRD for the decision calculate room price linked to Create Offer in Figure 4 above. This decision "Room Price" takes into consideration the type of room chosen by the client, the check-in date, number of people expected to stay in the room, and the basic room price. Its decision logic is represented in Table 2 which specify each output value (Room Price) in each scenario to which the decision is applied by using its inputs.

	Inputs				Outputs
U	Room	Check-in Date	Basic Room Price	Number of People	Room Price
	{16 SQ METERS,25 SQ METERS,40 SQ METERS}	{WORKDAY,WEEKEND}	Currency (€)	[1..4]	Currency (€)
1	= 16 SQ METERS	= WORKDAY	= 100 €	= 1	100 €
2	= 16 SQ METERS	= WORKDAY	= 100 €	= 2	110 €
3	= 16 SQ METERS	= WEEKEND	= 110 €	= 1	110 €
4	= 16 SQ METERS	= WEEKEND	= 110 €	= 2	120 €
5	= 25 SQ METERS	= WORKDAY	= 110 €	= 1	110 €
6	= 25 SQ METERS	= WORKDAY	= 110 €	= 2	120 €
7	= 25 SQ METERS	= WEEKEND	= 120 €	= 1	120 €
8	= 25 SQ METERS	= WEEKEND	= 120 €	= 2	130 €
9	= 40 SQ METERS	= WORKDAY	= 120 €	= 1	120 €
10	= 40 SQ METERS	= WORKDAY	= 120 €	= 2	130 €
11	= 40 SQ METERS	= WORKDAY	= 120 €	= 3	150 €
12	= 40 SQ METERS	= WORKDAY	= 120 €	= 4	170 €
13	= 40 SQ METERS	= WEEKEND	= 130 €	= 1	130 €
14	= 40 SQ METERS	= WEEKEND	= 130 €	= 2	140 €
15	= 40 SQ METERS	= WEEKEND	= 130 €	= 3	160 €
16	= 40 SQ METERS	= WEEKEND	= 130 €	= 4	180 €

Table 2. Decision Table. Decision: Calculate Room Price.

Figure 5 and Table 2 illustrate the decision requirements and decision logic for the decision to Calculate Room Price but may not provide a clear understanding of the decision and will not facilitate maintenance. This could be because too many details have been compressed in the models that could have been isolated to get a full understanding of the requirements of the decision and the decision logic. Multiple sub decisions have been compressed into a single decision because of the desire to see all the logic in one place. This obscure the individual decisions by mixing them makes reuse harder and discourages independent evolution according to Taylor and Purchase (2016).

To simplify and improve the representation of Table 2, OMG supports that adjacent input entry cells from different rules with the same content and same (or no) prior cells can be merged, but rule output cells cannot be merged except for crosstabs. With this consideration, Table 2 can be better represented in Table 3. Table 3 is just another representation of Figure 6 and is more friendly and avoid repetition of input entry, but does not change entries.

Room Price					
U	Room *("16 SQUARE METERS","25 SQUARE METERS","40 SQUARE METERS")*	Check-in Date *("WORKDAY","WEEKEND")*	Basic Room Price CURRENCY (EURO)	Number of people (1-4)	Room Price CURRENCY(EURO)
1				1	100
2		*"WORKDAY"*	100	2	110
3				1	110
4	*"16 SQUARE METERS"*	*"WEEKEND"*	110	2	120
5				1	110
6		*"WORKDAY"*	110	2	120
7				1	120
8	*"25 SQUARE METERS"*	*"WEEKEND"*	120	2	130
9				1	120
10				2	130
11				3	150
12		*"WORKDAY"*	120	4	170
13				1	130
14				2	140
15				3	160
16	*"40 SQUARE METERS"*	*"WEEKEND"*	130	4	180

Table 3. Decision Table. Decision: Calculate Room Price.

To enhance comprehension and maintainability of decision models, as well as, to avoid obscuring individual decisions, encourage independent decision evolution and complexity, decision models could be refactored. According to B. Weber et al (2011), primarily impact internal quality and ensure concept integrity, and foster maintainability. D. Calvanese et al (2018) argued that a common refactoring pattern is to simplify a Decision Table by merging pairs of rules into a single more general rule.

4.4.2 Divide and Conquer Pattern.

According to Taylor and Purchase (2016), the divide and conquer pattern is a means of splitting a single, over-complicated decision (Swiss and knife decision) that violates complex best practices into sub decisions that together perform the same tasks. To them, this pattern splits Decision Tables into subordinates of much smaller and manageable sizes using a partition that preserves behaviour. The nature of the partition depends on the original table and the motivation for applying this pattern. Taylor and Purchase (2016) specified that the split into sub tables must ensure mutually independent sub-tables that are unlikely to require coordinated maintenance and reduce sparsity.

When a Decision Table is split into fragments using the divide and conquer pattern, the results have to be consolidated by the use of a single parent decision. This parent can be an *Active* parent that select outcomes from the sub-tables to use the overall result. A *Passive* parent that allows the outcomes of the split tables to determine the result, typically by consulting each in priority sequence. Each sub-table uses a special return value to signify *I return no outcome of value* that causes the parent to consult other children.

Bruce Silver (2016) called this pattern "Action Sub-table Pattern". Applying the divide and conquer pattern or action sub-table pattern to Figure 5 and Table 3 will result in the following:

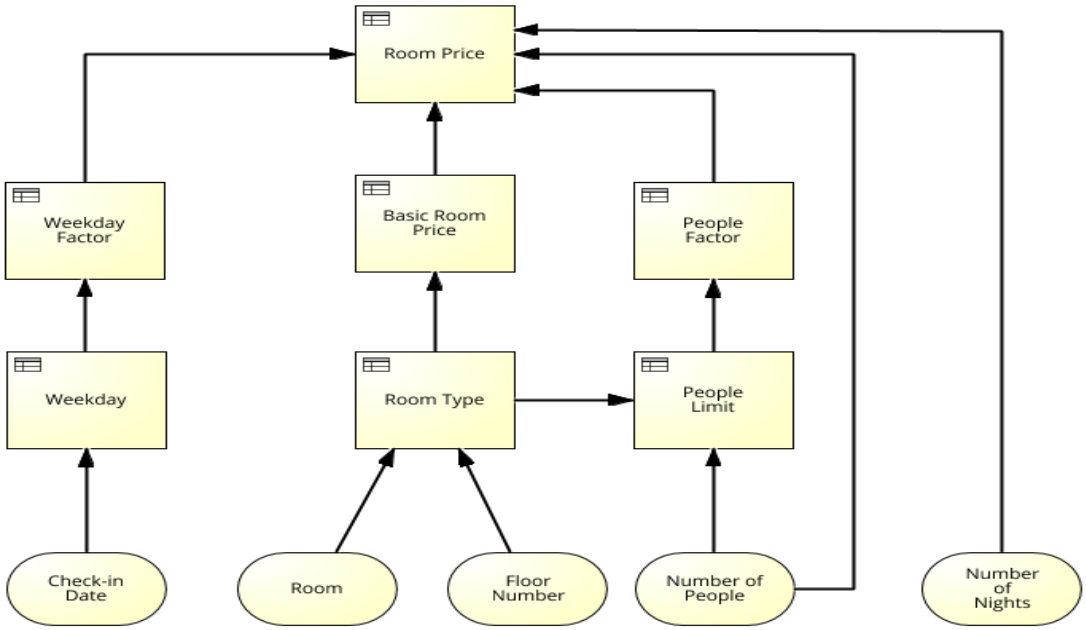

Figure 6. DRD. Decision: Calculate Room Price.

Figure 6 is the refactored model of Figure 5 showing with the decision Calculate Room Price and showing sub decisions Weekday Factor, Basic Room Price, People Factor, Weekday, Room Type, and People Limit.

Room Price	
WeekdayEffect	*BasicRoomPrice*WeekdayFactor*
*(BasicRoomPrice+WeekdayEffect+PeopleEffect)*NumberOfNights*	

Table 4. Decision Table. Decision: Calculate Room Price.

Table 4 replaces table 2 and show a straightforward calculation of room price based on the basic room price, weekday, number of people and number of nights.

	Inputs	Outputs
U	Weekday	Weekday Factor
	{WORKDAY,WEEKEND}	Currency (€)
1	= WORKDAY	0 €
2	= WEEKEND	10 €

Table 5. Sub-Table. Decision: Weekday Factor. Categorizes the factor impacting the room price based on the weekday.

U	Inputs Check-in Date	Outputs Weekday
	Number	*{WORKDAY,WEEKEND}*
1	= 1	WORKDAY
2	= 2	WORKDAY
3	= 3	WORKDAY
4	= 4	WORKDAY
5	= 5	WEEKEND
6	= 6	WEEKEND
7	= 7	WORKDAY

Table 6. Sub-table. Decision: Weekday. Categorizes the booking depending on the day of the week.

U	Inputs Room	Inputs Floor Number	Outputs Room Type
	{16 Sq Meters,25 Sq Meters,40 Sq meters}	*[1...3]*	*{Speculaars,Jenever,Mode,Japanese,Harzalaar}*
1	= 16 Sq Meters	= 1	Speculaars
2	= 25 Sq Meters	= 1	Jenever
3	= 16 Sq Meters	= 2	Mode
4	= 25 Sq Meters	= 2	Japanese
5	= 40 Sq meters	= 3	Harzalaar

Table 7. Sub-table. Decision: Room Type. Categorizing the room type which is mainly determined by the size of the room.

U	Inputs Room Type {Speculaars,Jenever,Mode,Japanese,Harzalaar}		Outputs Basic Room Price *Currency (€)*
1	=	Speculaars	100 €
2	=	Jenever	110 €
3	=	Mode	100 €
4	=	Japanese	110 €
5	=	Harzalaar	120 €

Table 6. Sub-table. Decision: Basic Room Price. Categorizes the basic room price which is solely determined by the room type.

U	Inputs People Limit *Number*		Outputs People Factor *Currency (€)*
1	=	2	10 €
2	=	3	30 €
3	=	4	50 €

Table 6. Sub-table. Decision: People Factor. A factor impacting the room price based on the number of people to be accommodated in the room.

U	Inputs Number of People *Number*		Inputs Room Type {Speculaars,Jenever,Mode,Japanese,Harzalaar}	Outputs People Limit *Number*
1	-	=	Speculaars	2
2	-	=	Jenever	2
3	-	=	Mode	2
4	-	=	Japanese	2
5	-	=	Harzalaar	4

Table 7. Sub-table. Decision: People Limit. Categorizing the required limit of people depending on the particular type of room.

In refactoring Figure 5 and Table 2, although more decisions have been added to Figure 5, refactoring has enabled a dense Decision Table to be split into simpler, separate, and cooperating sub-decisions each of which addresses a separate aspect of the requirement of the main decision. The main motivation was to enhance comprehension, maintainability, and independent evolution of the decisions while avoiding complexity. It is apparent that the major benefits of this pattern are of two-fold; the single, complex decision can be translated to smaller and easier to understand tables which can be more easily maintained separately. The logic associated with each conclusion category can now evolve independently.

4.4.3 Exception Based Logic Pattern.

According to Taylor and Purchase (2016), this type of refactoring decision pattern is required in special cases. For example, special cases exist when some decisions present fairly simple logic for 98% of the data cases presented to them, but 98% of the effort of analysing and expressing the decision is taken up with the remaining 2% of cases. This pattern allows the separation of these two cases so that the former is not obscured by the latter and maybe independently reused. The structure of Decision Requirements Models (and often the associate Decision Logic Models) is dominated by special cases that occur in a tiny fraction of the presented data and which keep changing. Pervasive special cases should be treated with suspicion. Because of the rarity of expertise of these special cases, they can evolve into "magic" decisions that are so convoluted and poorly understood that modellers fear to change them. This type of decision model becomes hard to understand and accumulate stale rules.

To them, this pattern requires that the decision model be dominated by decisions (and decision logic) of the mainstream cases and that exceptional cases be specified and handled "separately" with a purpose-built, localized decision and then reintegrated with the overall outcome. This can be achieved in two ways depending on the structure of the decision model:

- Mainstream plus low-frequency exceptional cases. If the exceptional case (s) account for a very low volume of outcomes and everything else results in one of a reasonable number of mainstream outcomes, the exceptional cases should be factored out and made the focus of their own decisions within the same DRD. This is "separation."
- No mainstream, just a diverse set of similar frequency special cases. If the decision model consists of no mainstream cases and is entirely composed of a set of special cases, each of which accounts for a similar fraction of the data and has unique logic, then the model should be "partitioned" into entirely different decisions. A separate decision determines which partition should carter for each set of inputs.

Separation involves isolating the dependencies on Input Data and Knowledge Sources that are unique to a special case, to a single decision or a small network of decisions. The original, top-level decision should then have an information requirement directly to the top-level special case decision to allow the latter to "exercise its veto." Partitioning involves designing entirely separate DRDs for each of the special cases. These can pursue entirely different decision structures. If some of the special cases are similar to one another, a partition can feature more than one and use separation to isolate the special variations (Taylor and Purchase, 2016).

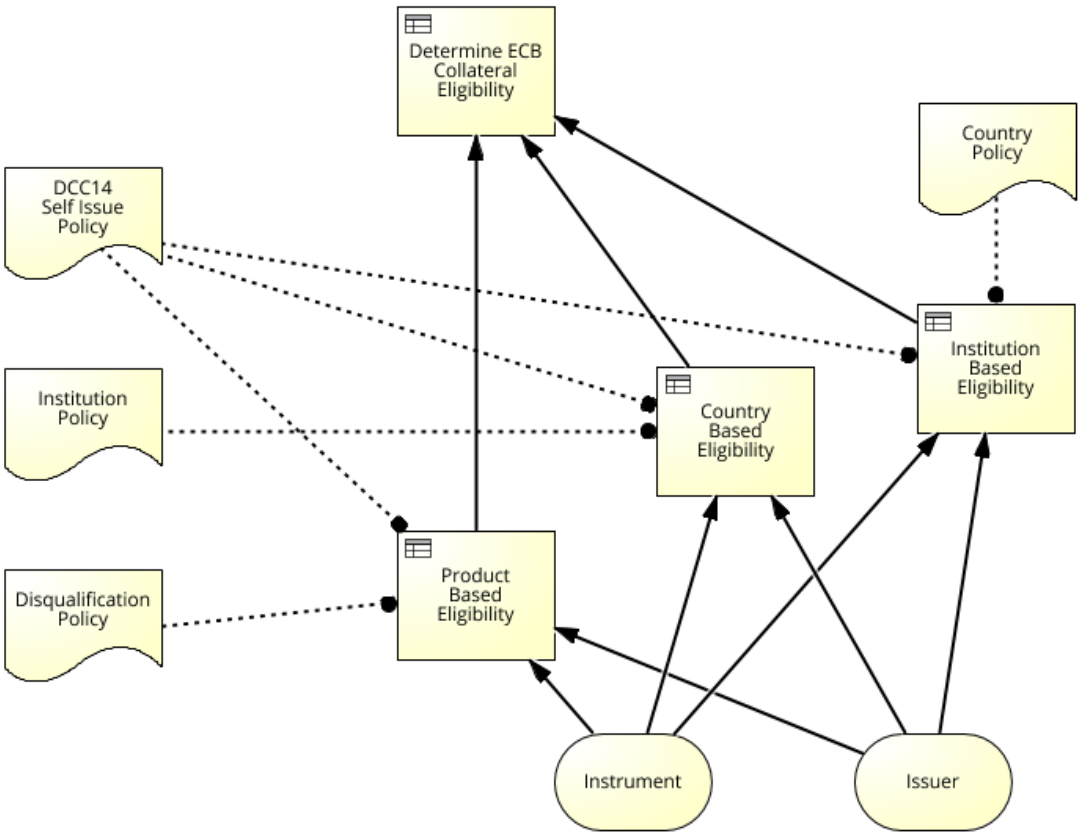

Figure 7. DRD with Widespread Exception
Source: Taylor and Purchase (2016)

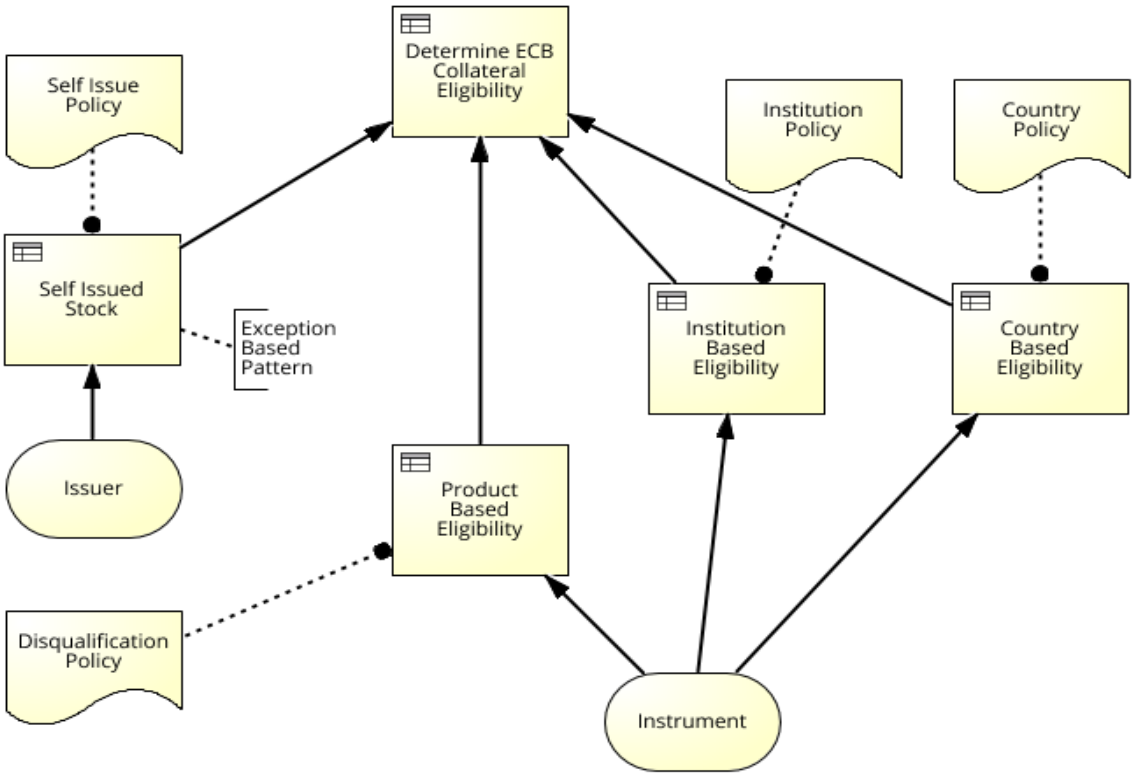

Figure 8. DRD with Localized Special Case
Source: Taylor and Purchase (2016)

Figure 7 illustrates a decision to determine if *instrument* can be used as collateral (*Determine ECB collateral Eligibility*). In this decision, it is important to handle the rare case in which a proposal to use an *instrument* as collateral must veto because it is associated with the organization requiring collateral(a company cannot use its stock as collateral). In the decision model, this exceptional veto influences the treatment of instruments of particular types (Product Based Eligibility), intermediaries (Institution Based Eligibility), and countries (Country Based Eligibility). The self-issue exceptional case (as supported by the Knowledge Source *DCC14 Self Issue Policy* and *Issuer* Data Input) has many incoming dependencies. This special case has far too much impact on the entire decision and it has not been sufficiently localized.

Figure 8 shows a much better DRD with the same behaviour as in figure 7. But the self-issue special case has been assigned to an explicit decision and this alone has a dependency on *Issuer* and *DCC14 Self Issue Policy*. This special case has been effectively isolated to the left of the diagram. Once this is achieved, maintenance of the self-issue decision is easier and the other decision are simplified.
With Exception Based Logic pattern, specialized expertise can be focused on smaller sub-parts of the decision model and changes are likely to have a more localized effect. It adds decisions to the decision model but reduces the model's overall complexity and ease comprehension. Special cases are prone to

change and obsolescence without warning. It is important to ensure that special case decisions are reviewed regularly and ruthlessly pruned of redundant content. The pattern is DRD oriented rather than a Decision Table.

4.4.4 Positive Validator Pattern

This pattern is also a recommendation of Taylor and Purchase (2016) and its complete description is from the authors.

With this positive validator pattern, a data structure must be checked for intra and inter-field validity in a compact manner that does not yield a wide Decision Table and also serves to document the data validation and make it as transparent as possible. A decision is required that identifies relevant flaws with inbound data. A parent decision may then reject data with serious flaws, protecting *pure* business decisions that consume this data from the need to do this, isolating them from the change in validation procedures and allowing them to focus on business issues.

Data validation frequently present three challenges. First, validation logic becomes mixed with business logic. This can cause decision models to become as much concerned with finding inconsistencies in the Input Data as they are with business logic. Secondly, validation logic is *fat*. Validation logic has many different conditions because many fields will typically require checking. Achieving a compact yet the readable representation of logic that has many conditions is challenging. Thirdly, validation decisions are often poor documentation of data validation. Often the business community looks to validation decisions not only to check its data but to act as documentation for data standards. That is self-documenting validation.

This pattern assumes that all attributes of an Input Data are individually available in a DMN context and that a context variable *attributes* contains a list of their names. Using this context variable allows DMN text rules to replace Decision Tables in which many columns would be needed if each were dedicated to specific attributes. The *ensure* conditions in this pattern use negative logic and are contrary to the conventions of DMN. Thus, the pattern's analysis is not detailed in this study and is not emphasized.

5. Conclusion, Limitation, and Recommendation.

5.1 Conclusion

This thesis is a structured literature review on DMN models with a stated purpose of investigating to identify existing DMN refactoring patterns that could be adopted to ensure that a poorly structured and difficult to maintain DMN model can be restructured to ease comprehension and maintenance without any effect on the model's semantics or external behaviour. DMN standard version 1.2 (2019) as prescribed by OMG was used as the basis for empirical investigation. The standard is for representing operational decisions and complements BPMN with a notation for modelling decision logic and dependencies between decisions and data elements. It is designed to be used alongside BPMN to create a bridge between business decision design and decision implementation.

DMN consist of two levels. The decision requirements level which is represented by the DRD and describe the requirements of decisions and the dependencies between elements involved in a decision model. And the decision logic level which describes the actual decision logic applied to take the decision and is represented by the Decision Table (Boxed Expression in this study). The two levels can either be used independently or in conjunction without referencing any business process to model a decision. DRD could be used independently to model human decisions. And the decisions would probably be described at quite a high level using natural language rather than decision logic. For fully automated decisions, there is more emphasis on detailed decision logic and the decisions could be modelled using Decision Table independent of a DRD. However, for partially automated decisions that involved both human and automated decisions, DRD and Decision Table are used in conjunction for a complete decision model.

Modelling a decision using DMN whether, in DRD or Decision Table, or both, there are style rules and convention standards that could be used to ensure a declarative, complete, and consistent decision model. The DMN specification provides standards that should be used but also give room for other styles and conventions (best practices) that could enhance understanding of DMN decision models. No matter the acceptable style and methodology used, it has to be adopted throughout the model to ensure consistency. Decision models formalize decision making to make it clear and widely understood, managed, and be used effectively. Conditional expressions should be mutually exclusive and complete, regardless of how they are modelled. Otherwise, the hit policy describes how they match with rules.

However, some DMN models could be challenging to read and complex. These challenges may be as a result of several reasons: repetition of conditions, obscured individual decisions, hard to reuse individual decisions, and unacceptable standards used, etc. as a result, the models have to be refactored so that these problems could be resolved. We identified three refactoring patterns that could be adopted. The choice of any chosen refactoring pattern would depend specifically on the

particular modelling challenge propelling refactoring. Regardless of the chosen pattern, it should be capable of enhancing understandability and foster maintainability through behaviour preserving.

5.2 Limitation

The scope of this study was limited to DMN Models. This was the major limitation of this study and which significantly influenced the extension of the length of time set for the search of existing literature because of the difficulty in identifying related publications. As a result, extensive search and reading were required, and out of the over 70 papers identified, only 3 recommended some decision refactoring patterns that can be adopted by modellers. The lack of additional finance to obtain more publications which often were not related to this study, though already paid for, was also a barrier.

5.2 Recommendation.

Generally, DMN is gaining increasing recognition within the modelling community. The main focus of this thesis was to identify DMN refactoring patterns that could be applied to restructure DMN models as regard improving understanding and maintenance of the decision models. We found very limited theoretical literature (three authors) on DMN refactoring patterns, and limited literature as well on DRD and Decision Table dependence and independence. As a result, we recommend further research in deriving more possible refactoring patterns that decision modellers could adopt to cope with the enormous modelling challenges faced in restructuring decision model representation to foster understanding, reuse, and maintenance of decision models. As well as further investigations on identifying possible situations that require DRD and Decision Table dependence and independence.

References

Balakrishnan, N., Render, B., Stair, R. M., & Munson, C. (2007). Managerial decision modeling.

Batoulis, K., Meyer, A., Bazhenova, E., Decker, G., & Weske, M. (2015, June). Extracting decision logic from process models. In *International conference on advanced information systems engineering* (pp. 349-366). Springer, Cham.

Blenko, M. W., Mankins, M. C., & Rogers, P. (2010). The decision-driven organization. *Harvard Business Review*, *88*(6), 54-62.

Bryman, A. (2016). *Social research methods*. Oxford university press.

Calvanese, D., Dumas, M., Laurson, Ü., Maggi, F. M., Montali, M., & Teinemaa, I. (2018). Semantics, analysis and simplification of DMN decision tables. *Information Systems*, *78*, 112-125.

Chenail, R. J. (2011). Ten Steps for Conceptualizing and Conducting Qualitative Research Studies in a Pragmatically Curious Manner. *Qualitative Report*, *16*(6), 1713-1730.

Christopoulou, A., Giakoumakis, E. A., Zafeiris, V. E., & Soukara, V. (2012). Automated refactoring to the strategy design pattern. *Information and Software Technology*, *54*(11), 1202-1214.

Cinnéide, MO (2001). *Automated application of design patterns: a refactoring approach* . Dublin: Trinity College.

Cronin, P., Ryan, F., & Coughlan, M. (2008). Undertaking a literature review: a step-by-step approach. *British journal of nursing*, *17*(1), 38-43.

Debevoise, T., Taylor, J., Sinur, J., & Geneva, R. (2014). *The MicroGuide to process and decision modeling in BPMN / DMN: building more effective processes by integrating process modeling with decision modeling* . CreateSpace independent publishing platform.

Demeyer, S., Ducasse, S., & Nierstrasz, O. (2002). *Object-oriented reengineering patterns*. Elsevier.

Deryck, M., Hasić, F., Vanthienen, J., & Vennekens, J. (2018, September). A case-based inquiry into the decision model and notation (DMN) and the knowledge base (KB) paradigm. In *International Joint Conference on Rules and Reasoning* (pp. 248-263). Springer, Cham.

Figl, K., Mendling, J., Tokdemir, G., & Vanthienen, J. (2018). What we know and what we do not know about DMN. *Enterprise Modelling and Information Systems Architectures (EMISAJ)*, *13*, 2-1.

Fowler, M. (2018). *Refactoring: improving the design of existing code*. Addison-Wesley Professional.

Ghlala, R., Aouina, Z. K., & Said, L. B. (2017, July). MC-DMN: Meeting MCDM with DMN involving multi-criteria decision-making in business process. In *International Conference on Computational Science and Its Applications* (pp. 3-16). Springer, Cham.

Guizzardi, R., Perini, A., & Susi, A. (2018, June). Aligning Goal and Decision Modeling. In *International Conference on Advanced Information Systems Engineering* (pp. 124-132). Springer, Cham.

Haendler, T., & Frysak, J. (2018). Deconstructing the Refactoring Process from a Problem-solving and Decision-making Perspective. In *ICSOFT* (pp. 397-406).

Hall, O. (2018). Business Decisions or Rules-Why not Both? The Views of Three Decision Modelling Experts.

Hasić, F., De Smedt, J., Broucke, S. V., & Asensio, E. S. (2020). Decision as a Service (DaaS): A Service-Oriented Architecture Approach for Decisions in Processes. *IEEE Transactions on Services Computing*.
Jackson, S. A., Kleitman, S., Stankov, L., & Howie, P. (2016). Decision pattern analysis as a general framework for studying individual differences in decision making. *Journal of Behavioral Decision Making*, *29*(4), 392-408.
Levy, Y., & Ellis, T. J. (2006). A systems approach to conduct an effective literature review in support of information systems research. *Informing Science*, *9*.
Mertens, S., Gailly, F., & Poels, G. (2015). Enhancing declarative process models with DMN decision logic. In *Enterprise, Business-Process and Information Systems Modeling* (pp. 151-165). Springer, Cham.
Model, B. B. P. Notation-http://www. omg. org/spec. *DMN/1.2/[retrieved: October, 2019]*.
Newing, H. (2010). *Conducting research in conservation: social science methods and practice*. Routledge.
Opdyke, WF (1992). Refactoring object-oriented frameworks.
Roberts, DB, & Johnson, R. (1999). *Practical analysis for refactoring* . University of Illinois at Urbana-Champaign.
Riehle, D., & Züllighoven, H. (1996). Understanding and using patterns in software development. *Tapos*, *2*(1), 3-13.
Silver, B. (2016). *DMN method and style: The practitioner's guide to decision modeling with business rules*. Cody-Cassidy Press.
Slavova, P. (2017). INTEGRATION OF "BUSINESS DECISION MODELING" IN COMPANIES FROM BULGARIAN REALITY. *Trakia Journal of Sciences*, *15*(1), 108-114..
Tang, Y. (2009). *On semantic decision tables* (Doctoral dissertation, PhD Thesis, Vrije Universiteit Brussel).
Tang, Y., & Meersman, R. (2008). Towards building semantic decision tables with domain ontologies. In *Challenges in Information Technology Management* (pp. 24-30).
Taylor, J., & Purchase, J. (2016). *Real-world decision modeling with DMN*. Tampa: Meghan-Kiffer Press.
van Der Aalst, W. M., Ter Hofstede, A. H., Kiepuszewski, B., & Barros, A. P. (2003). Workflow patterns. *Distributed and parallel databases*, *14*(1), 5-51.
Vanthienen, J., & Wets, G. (1994). From decision tables to expert system shells. *Data & Knowledge Engineering*, *13*(3), 265-282.
von Halle, B., & Goldberg, L. (2009). The Decision Model# 1: Linking Business Leaders and Technology.
Von Halle, B., & Goldberg, L. (2010). The decision model# 2: Improving process models and the requirements process. *Knowledge Partners International, LLC*.
Weber, B., Reichert, M., Mendling, J., & Reijers, H. A. (2011). Refactoring large process model repositories. *Computers in Industry*, *62*(5), 467-486.

Webster, J., & Watson, R. T. (2002). Analyzing the past to prepare for the future: Writing a literature review. *MIS quarterly*, xiii-xxiii.

https://editor.signavio.com/p/editor?id=b83f99147bb648b39fef4aa8eb525bb7 Table 2 DRD. Room Price .

https://editor.signavio.com/p/editor?id=0daa82c7a0594fe5bda4aa305b9ec34b Figure4 BPMN subprocess. Create Offer Subprocess.

https://editor.signavio.com/p/editor?id=c6dbb39b3f504c4dbe3de124b6a0f9cc Figure 3 BPMN Process. Book Hotel Room.

https://editor.signavio.com/p/editor?id=bec8b3fac7754861afa325d8808432c2 Figure 6 DRD. Room Price refactored.