Change Patterns for Decision Model and Notation (DMN) Model Evolution

Faruk Hasić and Estefanía Serral
Research Centre for Information Systems Engineering (LIRIS)
KU Leuven, Brussels, Belgium
{estefania.serralasensio, faruk.hasic}@kuleuven.be

Abstract

Information systems rely on decision knowledge during their execution. A recently introduced standard, the Decision Model and Notation (DMN), has been adopted in both industry and academia as a suitable method for modelling decision knowledge. However, this decision knowledge is not static and may undergo changes after system deployment. DMN change patterns have not yet been studied in the literature. This paper fills this gap by presenting an initial set of DMN change patterns. The patterns presented in this paper will not only facilitate the understanding of decision change management, but can also be capitalised on for, among other things, adapting decision management systems to be more flexible, consistency checking of decision models, and developing modelling tools that facilitate those changes.

Index terms— Decision Model and Notation (DMN), Model Evolution, Change Patterns

1 Introduction

Decision Model and Notation (DMN) is a recently introduced decision modelling standard that has enjoyed significant interest in literature [1, 2, 3, 4, 5]. DMN consists of two levels that are to be used in conjunction. First, the decision requirement level represented by the Decision Requirement Diagram (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. The DMN standard employs rectangles to depict decisions and sub-decisions, and ovals to represent data input. The decision logic level is usually represented in the form of decision tables. An example of a DRD diagram deciding on the severity of Chronic Obstructive Pulmonary Disease (COPD) in a patient is provided in Figure 1, while Figure 2 depicts the top-level decision logic in the form of a decision table.

Although it is obvious that the reasoning of decisions can change over time to adapt to changing requirements, all current works approach DMN from a static perspective, i.e., no attention has been given to changing or adaptable decision models. In this paper we approach this research gap by identifying a first set of change patterns that can be applied to DMN decision models. These findings can aid in developing and implementing dynamic decision management tools and systems that meet the requirements of flexible and changing decision knowledge.

This paper is structured as follows. Section 2 constitutes a related work section. In Section 3 runtime evolution of decision models is explained, while possible decision model change patterns are discussed in Section 4. Section 5 deals with future research. Finally, Section 6 concludes the paper.

2 Related Work

Most works around DMN have focused on the integration of process and decision models from a modelling point of view e.g., [2, 6, 7, 8, 9]. Others focus on the automatic discovery of decision model from enriched process event logs [10, 11]. Furthermore, literature provides a set of tools for modelling DMN models [12, 13, 14].

In: D. Di Nucci, C. De Roover (eds.): Proceedings of the 18th Belgium-Netherlands Software Evolution Workshop, Brussels, Belgium, 28-11-2019, published at http://ceur-ws.org

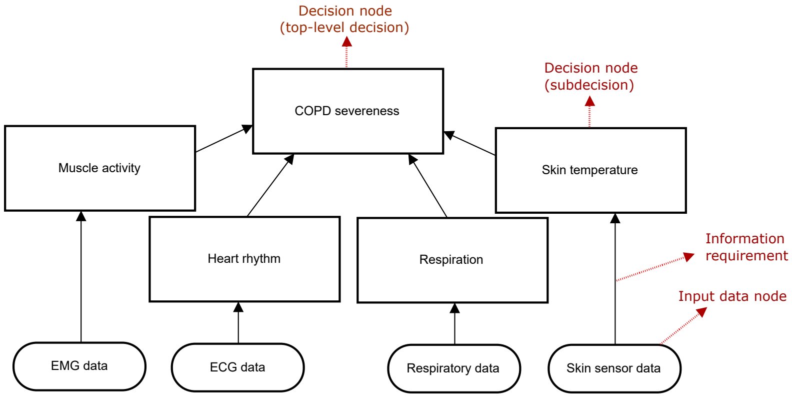

Figure 1: A DMN model for COPD severeness.

COPD Severeness

severeness

U	Input +				Output +
	respiration	skin temperature	heart rhythm	muscle activity	severeness
	string	string	string	string	string
1	"fast"	"cold"	"fast"	"normal","hyper"	"severe"
2	"fast"	"cold"	"normal"	"hyper"	"severe"
3	"fast"	"normal"	"fast"	"hyper"	"severe"
4	"fast"	"cold"	"normal"	"normal"	"mild"
5	"fast"	"normal"	"fast"	"normal"	"mild"
6	"fast"	"normal"	"normal"	"normal","hyper"	"mild"
7	"normal"	"normal","cold"	"normal","fast"	"normal","hyper"	"none"

Figure 2: A decision table for COPD severeness.

The ability of a knowledge-based system to efficiently deal with decision rule changes is considered a key property in literature [15, 16, 17, 18, 19, 20, 21, 22]. This way, business rigidity is avoided and the ability to transform the underlying business rules to new realities is facilitated. However, existing DMN decision model literature addresses decision modelling from a static perspective where decision models are built and used, without any form of model evolution. Change patterns are often used to define the possible evolution of models. These change patterns rely on the elementary edit operations that can be applied on the model elements, i.e., insertion and deletion, as well as substitution, which in essence is a combination of insertion and deletion [23]. Furthermore, change patterns can help facilitate the understanding of model change management as they provide a guide for implementing changes to models while maintaining model consistency. Nonetheless, change patterns for decision models were not yet addressed in the literature.

3 Runtime Decision Evolution

In this section we explain how decision models are used at runtime. An overview of decision model execution is given in Figure 3. In what follows, we take a closer look at the elements of the architecture and their interrelations.

Decision models can be specified according to the DMN standard. The models are saved in a file repository as an Extensible Markup Language (XML) file, adhering to the XML Metadata Interchange (XMI) standard [24], as specified by DMN 1.2 [1]. Execution engines are capable of interpreting XML files of decision models from the file repository, and consequently executing the corresponding decisions. Such an engine is provided by Camunda [25]. The engine executes the decisions and consequently the software services related to the decisions. The engine logs the executions for possible future analyses.

The connection between the aforementioned components is shown in Figure 3. The arrows between the components represent information flows. Namely, the decision models are fed into the file repository, which is then fed to the execution engine for enactment. The engine executes the decisions and logs the execution.

Changing the decision model at runtime will result in a new XML specification of the model, i.e., a new model variant. Deploying a new model variant for interpretation affects all future instances, as they will follow the newest model variant. Old running instances will continue to run according to the old model specification as a result of the variant versioning mechanism of the engine. This is a straightforward approach to assure that old instances continue to run in a sound way.

4 Towards Decision Model Change Patterns

To accommodate decision model changes, designers should be able to evolve the decision models after deployment. A number of changes can occur in the decision model. The core elements of a decision model are depicted in Figure 1, i.e. the input data and the decision nodes within a DRD, connected via information requirements arrows. The logic encapsulated in a decision node is usually modelled with decision tables, such as shown in Figure 2. To determine the change patterns, we investigate the changes that can manifest themselves on the core DMN elements, provided in the meta-model of the DMN specification [1], at different levels of granularity. First, we assess the change patterns within a single decision rule, i.e., changing the inputs and outcomes of a single rule or row in the decision table. Next, we look at change patterns for a decision rule in its entirety, i.e., adding or deleting decision rules from a decision table. These change patterns all pertain to a single node of the DRD, i.e., a single decision table, according to the DMN decision table meta-model [1]. Finally, we investigate change patterns on the topological structure of the DRD itself, respectively the addition and deletion of decision nodes and data input nodes. The change patterns are derived from the formalisation of core decision model elements in [2] and the elementary edit operations that can be applied on the elements, i.e., insertion and deletion, as well as substitution, which in essence is a combination of insertion and deletion [23]. Table 1 provides an overview of the change patterns directly relating to core DMN elements.

4.1 Change patterns within decision rules

We indicate a change pattern with $\Delta\Pi$. For changes within decision rules, we can distinguish three elements in the decision table that can undergo changes: the inputs, the outputs, and the logic mapping the inputs to the outputs, i.e. the decision rules:

1. **$\Delta\Pi$*1*: Excluding a decision input** indicates deleting an existing input variable from a decision table. However, simply deleting an input variable can render the decision table to be incomplete and incorrect. As such, the table may need to undergo refactoring in order to render a decision table that is complete and correct [26].
2. **$\Delta\Pi$*2*: Including a decision input** indicates adding a new input variable to a decision table. Similar to $\Delta\Pi$*1*, the table may need to undergo refactoring after this change pattern is applied
3. **$\Delta\Pi$*3*: Excluding a decision output** indicates the deletion of an existing output from the output set of a decision table.
4. **$\Delta\Pi$*4*: Including a decision output** indicates an addition of a new output to the output set of a decision table.
5. **$\Delta\Pi$*5*: A decision logic change** indicates a change in relating the existing input symbols to the existing output symbols within the decision table, i.e., the mapping from inputs to outputs.

4.2 Change patterns on decision rules in their entirety

Next to changes within decision rules, we can view decision rules as an atomic entity and perform changes on the decision rule in its entirety:

1. **$\Delta\Pi$*6*: Excluding a decision rule** if a decision rule is deemed irrelevant at a certain point in time. The decision rule can be deleted in its entirety from a decision table. Notice that by deleting decision rules, the decision table may not be complete anymore. To avoid this, either the decision table should be completed by ensuring that all input values are mapped to existing decision outcomes, or the system should be redesigned to capture the possibility of no decision outcome being returned.
2. **$\Delta\Pi$*7*: Including a decision rule** if a new decision rule is deemed relevant at a certain point in time. The decision rule can be added in its entirety to an existing decision table.

Table 1: Overview of decision model change patterns.

Decision table change patterns	
Changes within decision rules	
$\Delta\Pi$*1*	Decision input exclusion.
$\Delta\Pi$*2*	Decision input inclusion.
$\Delta\Pi$*3*	Decision output inclusion.
$\Delta\Pi$*4*	Decision output exclusion.
$\Delta\Pi$*5*	Decision rule logic change.
Changes on decision rules	
$\Delta\Pi$*6*	Decision rule exclusion.
$\Delta\Pi$*7*	Decision rule inclusion.
Decision requirements diagram change patterns	
Decision node changes	
$\Delta\Pi$*8*	Decision node exclusion.
$\Delta\Pi$*9*	Decision node inclusion.
Input data node changes	
$\Delta\Pi$*10*	Input data node inclusion.
$\Delta\Pi$*11*	Input data node exclusion.

4.3 Change patterns on the decision nodes in the DRD

This subsection deals with the deletion or addition of decision nodes in the decision requirements diagram.

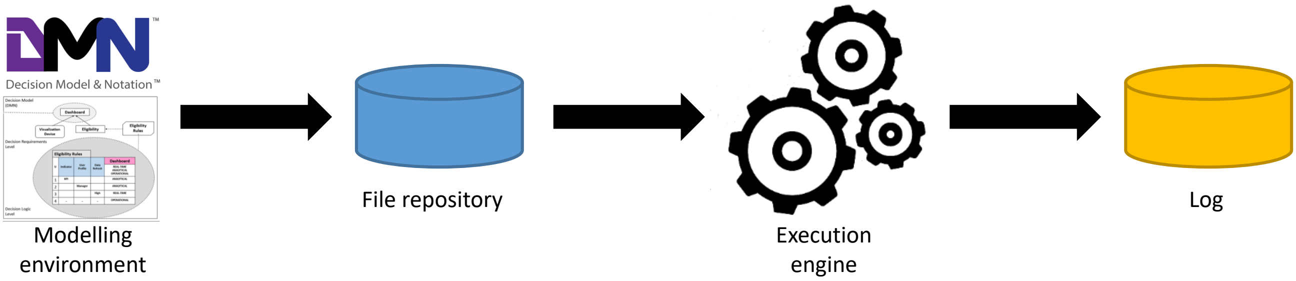

Figure 3: An overview of decision model execution.

1. $\Delta\Pi$*8*: **Excluding a decision node** from the DRD corresponds to deleting all decision rules from a decision node. Hence, this change pattern is an aggregation of multiple *exclude decision rule* changes ($\Delta\Pi$*6*). Note that deleting a decision node also deletes all its incoming and outgoing information requirements arrows.

2. $\Delta\Pi$*9*: **Including a decision node** to the set of decision nodes corresponds to adding a new decision table, and thus, adding multiple decision rules encapsulated in the decision node. Hence, this change pattern is in essence an aggregation of multiple *include decision rule* changes ($\Delta\Pi$*7*). Note that adding a decision node also adds the necessary incoming and outgoing information requirements arrows.

4.4 Change patterns on the input data nodes in the DRD

This subsection deals with the deletion or addition of input data nodes in the decision requirements diagram.

1. $\Delta\Pi$*10*: **Including an input data node** to the set of data input nodes also adds its necessary input requirement arrows and connects it to the relevant decision nodes in the DRD. Notice that this change pattern on the DRD level corresponds to adding a new input variable to the decision table that requires the newly added data input node. Hence, this change pattern is equivalent to $\Delta\Pi$*2*.

2. $\Delta\Pi$*11*: **Excluding an input data node** from the set of data input nodes also deletes all its input requirement arrows. Notice that this change pattern on the DRD level corresponds to deleting an input variable from the decision table that required the input data node. Hence, this change pattern is again equivalent to $\Delta\Pi$*1*.

Note that adopting a change pattern on a DMN decision model can lead to within-model inconsistencies and that additional change patterns may need to be propagated throughout the entire decision model to safeguard within-model consistency. For instance, deleting an input data node will also require deleting the inputs in the decision tables that require the deleted input data node.

5 Future Work

In future work, we will investigate how a change in the decision model can require the triggering of other changes in order to safeguard within-model consistency. Additionally, we will investigate how changing decision models impacts other systems and models that rely on the logic encapsulated in the decision models. More specifically, we will examine how the proposed change patterns influence process and decision model consistency in an integrated process and decision model environment. Changing the underlying decisions of a process can lead to change patterns in the process model as well if the sound interaction between the process and decision model is to be ensured.

Furthermore, flexible decision models are of particular interest to Internet-of-Things (IoT) process settings [27], as IoT process are inherently subject to a dynamic and changeable environment. Therefore, we will investigate how these change patterns manifest themselves in decision-intensive IoT processes.

Finally, we will look into developing a tool that provides possibilities for DMN model evolution with automated model consistency checking and repair, while maintaining the link with the Camunda execution engine.

6 Conclusion

This paper presents an initial set of decision model change patterns for the evolution of DMN decision models. We recognise that the adaptation of a DMN decision model can lead to within-model inconsistencies and that additional change patterns may need to be propagated throughout the entire decision model to safeguard within-model consistency.

References

[1] OMG. Decision Model and Notation (DMN) 1.2, 2018.

[2] Faruk Hasić, Johannes De Smedt, and Jan Vanthienen. Augmenting processes with decision intelligence: Principles for integrated modelling. *Decision Support Systems*, 107:1 – 12, 2018.

[3] Faruk Hasić and Jan Vanthienen. From decision knowledge to e-government expert systems: the case of income taxation for foreign artists in belgium. *Knowledge and Information Systems*, Oct 2019.

[4] Faruk Hasić and Jan Vanthienen. Complexity metrics for dmn decision models. *Computer Standards & Interfaces*, 65:15 – 37, 2019.

[5] Marjolein Deryck, Faruk Hasić, Jan Vanthienen, and Joost Vennekens. A case-based inquiry into the decision model and notation (dmn) and the knowledge base (kb) paradigm. In Christoph Benzmüller, Francesco Ricca, Xavier Parent, and Dumitru Roman, editors, *Rules and Reasoning*, pages 248–263, Cham, 2018. Springer International Publishing.

[6] Faruk Hasić, Johannes De Smedt, and Jan Vanthienen. Redesigning processes for decision-awareness: Strategies for integrated modelling. In *International Conference on the Quality of Information and Communications Technology*, 2018.

[7] Faruk Hasić, Johannes De Smedt, and Jan Vanthienen. Developing a modelling and mining framework for integrated processes and decisions. In Christophe Debruyne, Hervé Panetto, Georg Weichhart, Peter Bollen, Ioana Ciuciu, Maria-Esther Vidal, and Robert Meersman, editors, *On the Move to Meaningful Internet Systems. OTM 2017 Workshops*, pages 259–269, Cham, 2018. Springer International Publishing.

[8] Thierry Biard, Jean-Pierre Bourey, and Michel Bigand. Dmn (decision model and notation): De la modélisation à l'automatisation des décisions. In *INFORSID 2017*, 2017.

[9] Kimon Batoulis, Anne Baumgraß, Nico Herzberg, and Mathias Weske. Enabling dynamic decision making in business processes with dmn. In *International Conference on Business Process Management*, pages 418–431. Springer, 2015.

[10] Júlio Campos, Pedro Richetti, Fernanda Araújo Baião, and Flávia Maria Santoro. Discovering business rules in knowledge-intensive processes through decision mining: an experimental study. In *International Conference on Business Process Management*, pages 556–567. Springer, 2017.

[11] Johannes De Smedt, Faruk Hasić, Seppe K.L.M vanden Broucke, and Jan Vanthienen. Holistic discovery of decision models from process execution data. *Knowledge-Based Systems*, 183:104866, 2019.

[12] Carl Corea and Patrick Delfmann. A tool to monitor consistent decision-making in business process execution. *Proceedings of the Dissertation Award, Demonstration, and Industrial Track at BPM*, pages 9–14, 2018.

[13] Knut Hinkelmann, Kyriakos Kritikos, Sabrina Kurjakovic, Benjamin Lammel, and Robert Woitsch. A modelling environment for business process as a service. In *International Conference on Advanced Information Systems Engineering*, pages 181–192. Springer, 2016.

[14] Knut Hinkelmann, Arianna Pierfranceschi, and Emanuele Laurenzi. The knowledge work designer-modelling process logic and business logic. In *Modellierung (Workshops)*, pages 135–140, 2016.

[15] Zohra Bellahsene. Schema evolution in data warehouses. *Knowledge and Information Systems*, 4(3):283–304, 2002.

[16] Natalya F Noy and Michel Klein. Ontology evolution: Not the same as schema evolution. *Knowledge and information systems*, 6(4):428–440, 2004.

[17] Wan MN Wan Kadir and Pericles Loucopoulos. Linking and propagating business rule changes to is design. In *Information Systems Development*, pages 253–264. Springer, 2005.

[18] Jérôme Boyer and Hafedh Mili. Agile business rule development. In *Agile Business Rule Development*, pages 49–71. Springer, 2011.

[19] Flavio Corradini, Alberto Polzonetti, and Oliviero Riganelli. Business rules in e-government applications. *arXiv preprint arXiv:1802.08484*, 2018.

[20] Gordon Blair, Nelly Bencomo, and Robert B France. Models@ run. time. *Computer*, 42(10):22–27, 2009.

[21] Michael Szvetits and Uwe Zdun. Systematic literature review of the objectives, techniques, kinds, and architectures of models at runtime. *Software & Systems Modeling*, 15(1):31–69, 2016.

[22] Sihem Loukil, Slim Kallel, and Mohamed Jmaiel. An approach based on runtime models for developing dynamically adaptive systems. *Future Generation Computer Systems*, 68:365–375, 2017.

[23] Andreas Wombacher and Maarten Rozie. Evaluation of workflow similarity measures in service discovery. *Service Oriented Electronic Commerce*, 80:51–71, 2006.

[24] OMG. Xml metadata interchange (XMI) 2.5.1, 2015.

[25] Camunda. Process engine. https://docs.camunda.org/manual/7.8/user-guide/process-engine/, 2018. Accessed: 2018-11-16.

[26] Diego Calvanese, Marlon Dumas, Ülari Laurson, Fabrizio M Maggi, Marco Montali, and Irene Teinemaa. Semantics, analysis and simplification of dmn decision tables. *Information Systems*, 2018.

[27] Faruk Hasić and Estefanía Serral Asensio. Executing IoT processes in BPMN 2.0: Current support and remaining challenges. *IEEE RCIS 2019 proceedings*, 2019.