Decision-Aware Information Systems

A Systems Modelling Perspective Bridging Decisions and Processes

Dissertation presented to obtain the degree of Doctor of Philosophy

by

Faruk HASIĆ

Number 692 May 2020

Committee

Chairperson	Prof. dr. Martina Vandebroek	KU Leuven
Supervisor	Prof. dr. Estefanía Serral Asensio	KU Leuven
Co-Supervisor	Prof. dr. Monique Snoeck	KU Leuven
	Prof. dr. Yves Wautelet	KU Leuven
	Prof. dr. Patrick Delfmann	University of Koblenz-Landau
	Prof. dr. Johannes De Smedt	University of Edinburgh

Acknowledgments

"No book can ever be finished. While working on it we learn just enough to find it immature the moment we turn away from it."

The Open Society and Its Enemies, 1950
— *Karl Popper*

To obtain a PhD one must tread a long and winding road. It is often said that the pursuit of a PhD should come with a government health warning. Fortunately, the long and winding road to a PhD is partially paved with the widespread encouragement of so many. In this section I hope to recognise the support I received during the past three years and eight months and to adequately convey my gratitude for the same.

First and foremost, I would like to thank my supervisor prof. dr. Estefanía Serral for the active support and care during this journey, the insightful questions and suggestions, and all the unconditional trust and confidence she placed in me. Fani, thank you for the many talks we had and for the many laughs we shared. I am proud to be your first PhD graduate and I am sure that many more will follow. Perhaps they will be a bit less stubborn than me. We will see.

Next, I wish to thank my co-supervisor prof. dr. Monique Snoeck. I am still amazed by your unrivalled expertise and insights which always surface when providing comments, suggestions, and questions that invariably go the core of the problem at hand. I like to think of you as the wise oracle of the group that always knows what is coming next and what course of action is best. Thank you for being there for the PhD candidates of the group, not only in terms of research support, but also for considering the well-being of us all.

Let me also thank the remaining members of the examination committee. My gratitude to prof. dr. Martina Vandebroek. Four years ago, you were present at my Master's thesis defence as a third reader and now you are closing the circle by chairing the examination committee of my PhD.

Furthermore, many thanks go to prof. dr. Yves Wautelet for always being open and ready to help, as well as for constantly looking out for and sharing possible future opportunities.

I am also grateful for the support of prof. dr. Patrick Delfmann. Thank you for enabling the teamwork between our research groups and for providing me and your PhD students with the trust and freedom to set up a smooth and efficient collaboration.

The last, but not least, examination committee member is prof. dr. Johannes De Smedt. Johannes, thank you for all the support, insights, talks, jokes, and laughs we shared in the past years. In a sense, you have been with me on this journey from the beginning. My time in Edinburgh with you, Ynte, Kasper, and Quincy will be something that I will always fondly remember. I am glad you have expanded your family with an additional information systems researcher. I wish all of you the best, except maybe Quincy because of his attempt to murder me in my sleep. Thank you for hosting me, for showing me your troublesome YouTube video taste, and for writing and recording our very first information systems research song *Decisions on my mind* which, if released, would definitely top the charts in Nerdville.

Special thanks go to prof. dr. Jan Vanthienen for giving me the opportunity to embark on the adventure of research, for introducing me to the compelling topic of processes and decisions, and for supporting me during the elaboration of Part II of this dissertation. Thank you for always organising the best team building activities and for sharing your wise quotes on research with us. One of my favourites that I remember is: *if you conduct a survey about nuclear physics, that does not make you an expert on nuclear physics.*

Furthermore, I would like to express my gratitude to the other faculty members of our research group. Professor Jochen De Weerdt, thank you for being so approachable and helpful, as well as

for always providing humorous insights into the world of academia during our many coffee breaks. I admire you for being able to successfully combine a strong career in academia and a busy, but undoubtedly fulfilling, family life. Thank you for providing me with the opportunity to conduct a research stay in Chile, which was truly an amazing experience. Professor Bart Baesens, thanks sharing your love for pizza fantasia, now I am a fan too. I enjoyed your funny stories and jokes at coffee breaks and receptions. We should have grabbed a drink together more often. Hopefully there will be time for that in the future. Prof. Seppe vanden Broucke, you are a talent like no other, with witty and funny insights into academia and beyond, which I enjoyed very much. Finally, professor Ferdi Put, I very much enjoyed our talks at the coffee machine, as well as your dry and subtle, yet sometimes direct remarks that always provoked laughter. Thank you for being the calm and friendly person that you are.

Now that I have addressed the elite of our research group, it is time to mention the plebs of academia, also known as PhD students. First, I will mention a few names of students that have graduated during my time. To the new PhD students in the group, these sound like names of heroes from tales of old. They are not really sure that those people ever existed, but their ghosts sometimes still roam the halls of our faculty. I have known many of them and I still have fond memories of our time together. Pieter, I mostly enjoyed your sense of shamelessly harsh humour. Michael, you were often a source of social cohesion in the group, or maybe you were just improvising, we will never know. Klaas, I enjoyed your company and your ability to be supportive of everyone. I feel that we should have spent more time together. Tine, I enjoyed your linguistic acrobatics which, to my surprise, is not inversely proportional to the amount of wine glasses you consume. Unfortunately, one cannot draw the same conclusions for Anthony's linguistic functionality and the amount of beer consumed. However, I did enjoy all the time you spent with us in this development cooperation project between the north and the south of the country. I hope you stay true to your Slavic roots and start learning Macedonian soon. Jenny, though it was short, it was fun having you around to provide a more exotic Cuban flavour to our group. María, I know you always hated my short jokes, but here comes a final one, I promise: I never knew Vikings could be that short. Tom, I know you never liked my jokes either, but someone had to point out the elephant in the room. Jasmien, I loved having you around because you always laughed at my jokes, but that is maybe because they were not about you. I will try to fix this in the future. Eugen, you helped a prejudice out of this world: you actually demonstrated that Germans have a sense of humour. Though the sample size is small, I will accept this as a proof by counterexample. Sandra, I would like to say that, like Eugen, you too helped a prejudice out of this world, namely that Montenegrins are lazy. I would like to say that, but I am still missing the proof by counterexample. I was always amazed at how we could speak two totally different languages and still understand each other. Together, I feel that we successfully introduced a bit of balkanisation in a civilised Western European institution.

Next to these veterans that graduated, I would like to mention my batch of PhD students, namely the batch that started at the end of 2016. Anthony was one of us, but then he decided to go back to the deep and dangerous south. Only myself, Galina, Daria, and Rongjia remained. A small batch of clueless junior PhD students. At first, we looked at the veterans that I mentioned before, and we thought to ourselves: will we ever be that advanced, that smart, and that capable? Then we realised those veterans were basically just faking it and we soon learnt to do the same. I was sceptical of Galina and Daria in the beginning because I come from a part of the world where the Russian influence, Russian spies, and the immanent threat of a Russian invasion were always feared. On top of that, for about two years the people in the research group could not tell which one of them was Daria and which one Galina. Surely, this was some KGB tactic that they employed to confuse us. But now, though we remain watchful all the time, we have learnt to accept them as one of our own. Rongjia was the spy sent by Beijing. She allegedly first came for a few months as an exchange student. Before we knew it, she occupied a permanent spot in the PhD group. In the beginning, Rongjia and I had a lot of fun together as we shared an office. Not just any office, the best and biggest corner office of our research group. Soon, the leaders of the research group decided that they should give that office to a more important and productive member of our group. So, we were kicked out and the coffee machine was awarded the corner office.

What was once a clueless batch of junior PhD students has become the senior group of PhD researchers. Soon, we saw new poor unfortunate souls enter the PhD life, like us back then, not knowing what they have started. It is remarkable how some of the new PhD students resemble the old ones. Jari reminds me of Pieter. They both think process mining and trace clustering are cool and they have a similar sense of humour. Björn sounds like María with a deeper voice, since Björn is actually a life-size Viking (sorry María, promise already broken). Carlos, my sincerest apologies for the stereotypical *Mexican sleeping next to a cactus with a sombrero on his head* jokes. I will try to decrease the frequency of those for a while, but they will surface again. Jeroen, always fun to complain to each other about which problems we encounter next with the refurbishment or building of our respective houses. Only we know the pain. Veda, it is fun to hear that Indian accent once again in real life. It reminds of my exchange to India years ago, when research was not even a thing that existed in my mind. Rafaël, fun to have another coke (the beverage) addict in the department. I felt alone for a long time. Steven, I still cannot believe you are that old. Ziboud, please, for the love of all that is holy, stop making those terrible puns and wordplay jokes. In fact, do humanity a favour and stop making any kind of jokes. We also have a few new colleagues at the Brussels campus. Pavani and Konstantinos are a true enrichment to our group. Sharing an office with them a few days a week has been truly enjoyable. I would also like to recognise my other office mates from different research groups for their company over the past years. Alex, it was a true pleasure being in the same office as you. Morteza, thank you for sharing your insights as a more experienced researcher. Talia, you are a funny and entertaining office mate. I do have the feeling that your presence in the office negatively correlates with my productivity. But at least we had a few good laughs.

Next to colleagues affiliated with the KU Leuven, I had the pleasure to collaborate with great researchers from all around the world. In particular, I would like to express my gratitude to Stephen Hasley of the American College of Obstetricians and Gynecologists and Letha Sooter of the University of Pittsburgh for involving my doctoral research in their clinical pathway modelling endeavours. Furthermore, I would like to thank Carl Corea of the

University of Koblenz-Landau for being open for our collaboration, for always being available for discussions, and for crunching the deadlines with me during the final part of my PhD research. I think that together we achieved quite a lot in a short period of time. I look forward to our future collaboration and I hope we meet each other soon in real life, preferably at a conference at an exotic location.

One of the highlights of the PhD trajectory was my research stay at the Complex Engineering Systems Institute of the Faculty of Physical and Mathematical Sciences of the University of Chile. I would like to thank the Web Science & Smart Technologies (WeSST) Lab for hosting me in Santiago and for including me in the group as one of their own. In particular, special thanks go to the head of the lab, prof. dr. Ángel Jiménez-Molina for his hospitality and for the many lunches and talks we shared during my stay. I will fondly remember my time at the lab and the office mates Cristian, Francisco, Pablo, Sebastián, and Macarena. Also many thanks to Marcelo for his kind attitude and helpful tips during my stay. Weónes Chilenos, you will be missed. I hope to see you all soon. Furthermore, thanks go to Beatriz and Giovanni for discussing future research with me while I was there. Of course when I think Chile, I think Luis Aburto Lafourcade. Luis, it was a great pleasure having you with us in Leuven and likewise a great pleasure seeing you again in Santiago. I would like to thank you and Paloma for your hospitality and for making time for the Leuven researchers while we were in Chile. I hope the two of you will have a long and happy life together and that we may meet again soon.

Above, I expressed my gratitude to people that I encountered as a result of my PhD track. I would also like to give a shout out to the many domestic and foreign family, friends, and housemates that have accompanied me along this journey, making the hardest parts more bearable.

Finally I would like to end by recognising the most important people in my life, my parents and my sister, to whom this PhD is dedicated. Thank you for all the support over all these years, in times both good and bad.

Faruk Hasić
May, 2020

To my parents,
and to my sister,
for their tremendous support.

Doctoral Dissertation Summary

Modern process-aware information systems heavily rely on decision knowledge for the development, execution, and evolution of operational processes. This decision knowledge drives process engineering, execution, and automation and impacts multiple process perspectives, such as the control flow, data, and resource perspectives. Therefore, a consistent integration of decisions and processes is imperative for a sound development and management of knowledge-based processes. This thesis presents advances in the field of process-aware information systems by augmenting the processes with a model-driven decision perspective. The contributions are divided into three parts.

The first part presents the integration of processes and decisions from a modelling viewpoint. By detecting inconsistencies in the integration of the models, principles and guidelines are proposed to ensure a consistent integration when modelling decision-aware processes or re-engineering processes for decision-awareness.

The second part capitalises on design principles from the service-oriented architecture paradigm to systematically address the separation of concerns between the process and decision perspectives, as well as their consistent integration. A layered reference design that takes these design principles into account is contributed and illustrated through decision-aware process discovery on a real-life enriched event log.

Finally, in the third part, change patterns for decision-aware processes are introduced, as well as prototype modelling tools that support the evolution of process and decision models through the change patterns. The tools comprise a modelling environment with built-in consistency verification capabilities, automated feedback, and user-driven automated mechanisms for restoring both within-model and between-model consistency.

In summary, this dissertation introduces a systems modelling perspective of decisions to process-aware information systems, thus rendering decision-aware processes. By doing so, it recognises the importance of decision models in model-driven software systems engineering.

Samenvatting Doctoraal Proefschrift

Moderne informatiesystemen steunen op beslissingskennis voor hun ontwikkeling, uitvoering en evolutie. Procesontwikkeling, uitvoering en automatisering worden vaak aangestuurd door beslissingen die meerdere perspectieven van het proces beïnvloeden, zoals de control flow, de data en de resources. Daarom is het consistent integreren van beslissingen en processen noodzakelijk voor een goede ontwikkeling en beheer van kennisgebaseerde processen. Dit proefschrift introduceert vorderingen in het gebied van procesbewuste informatiesystemen door de processen uit te breiden met een modelgestuurd beslissingsperspectief.

Het eerste deel behandelt de integratie van processen en beslissingen vanuit een modelleeringsoogpunt. Door inconsistenties in de integratie van de modellen te detecteren, worden principes en richtlijnen voorgesteld om een consistente integratie te waarborgen bij het modelleren van beslissingsbewuste processen.

Het tweede deel maakt gebruik van ontwerpprincipes uit het servicegeoriënteerde architectuurparadigma om systematisch de scheiding van belangen tussen het proces en het beslissingsperspectief te waarborgen, evenals hun consistente integratie. Een gelaagd referentieontwerp gesteund op deze ontwerpprincipes wordt voorgesteld en geïllustreerd door beslissingsbewuste procesontginning op een verrijkte event log.

In het derde deel worden veranderingspatronen voor beslissingsbewuste processen geïntroduceerd, alsook prototypes van modelleringstools die de evolutie van proces- en beslissingsmodellen

ondersteunen. De tools omvatten een modelleringsomgeving met consistentieverificatie, geautomatiseerde feedback en door de gebruiker aangestuurde geautomatiseerde mechanismen voor consistentieherstel, zowel binnen het model als tussen de modellen.

Aldus introduceert dit proefschrift een systeemmodelleringsperspectief van beslissingen voor procesbewuste informatiesystemen, resulterend in beslissingsbewuste informatiesystemen. Hiermee erkent dit proefschrift het belang van beslissingsmodellen in de modelgestuurde ontwikkeling van softwaresystemen.

Résumé de la Thèse de Doctorat

Les systèmes d'information modernes reposent sur la connaissance décisionnelle pour leur développement, exécution et évolution. L'ingénierie, l'exécution et l'automatisation des processus sont souvent guidés par des décisions qui ont un impact sur de multiples aspects du processus. Par conséquent, l'intégration cohérente des décisions et des processus est essentielle pour un développement et une gestion appropriés des processus basés sur la connaissance. Cette thèse présente les avancées dans le domaine des systèmes d'information centrés-processus en augmentant les processus avec une perspective de décision. Les contributions sont divisées en trois parties.

La première partie présente l'intégration des processus et des décisions du point de vue de la modélisation. En détectant les incohérences dans l'intégration des modèles, des principes et des recommandations sont suggérées pour assurer une intégration cohérente lors de la modélisation des processus centrés-décision ou de la réingénierie des processus afin qu'ils deviennent centrés-décision.

La deuxième partie s'appuie sur les principes de conception du paradigme de l'architecture orientée services pour traiter systématiquement la séparation des préoccupations entre les perspectives de processus et de décision. Une conception de référence à plusieurs niveaux qui tient compte de ces principes de conception est proposée et illustrée par la découverte de processus de prise de décision sur un journal d'événements réels.

Enfin, des schémas de changement pour les processus centrés-décisions sont présentés, ainsi que des prototypes d'outils de modélisation qui soutiennent l'évolution des processus et des modèles de décision grâce aux schémas de changement. Ces outils comprennent un environnement de modélisation avec des capacités intégrées de vérification de la cohérence, un mécanisme de feedback automatisé et des mécanismes automatisés orientés utilisateur pour restaurer la cohérence du modèle et entre les modèles.

En résumé, cette thèse introduit une perspective de décisions dans la modélisation des systèmes d'information centrés-processus. Ce faisant, elle reconnaît l'importance des modèles de décision dans l'ingénierie des systèmes logiciels guidée par les modèles.

Zusammenfassung der Doktorarbeit

Moderne Informationssysteme weisen derzeit im Rahmen ihrer Entwicklung, Ausführung und Weiterentwicklung einen enormen Bedarf an Entscheidungsfindungsmechanismen auf. Die Prozessentwicklung, -ausführung und -automation ist hierbei oft von Entscheidungsfindungsvorgängen getrieben, die sich auf diverse Aspekte des Prozesses wie etwa den Kontrollfluss, die Daten oder die Ressourcen auswirken können. In diesem Kontext ist die konsistente Integration von Entscheidungsfindungsmechanismen und Prozessen für eine fehlerfreie Entwicklung und Weiterführung von wissensbasierten Prozessen daher unumgänglich. Diese Arbeit präsentiert neue Erkenntnisse im Bereich der prozessbasierten ("process-aware") Informationssysteme, speziell der Erweiterung von Prozessen durch eine modellgetriebene Entscheidungsperspektive. Die Ergebnisse gliedern sich hierbei in drei Teile.

Der erste Teil untersucht die Integration von Prozessen und Entscheidungslogik aus dem Blickwinkel der Modellierungsperspektive. Durch die Identifikation von Inkonsistenzen zwischen Prozess- und Entscheidungsmodellen können so Heuristiken und Vorgehensmodelle vorgeschlagen werden, die eine konsistente Integration im Rahmen der Modellierung von entscheidungsbasierten Prozessen, bzw. der Verbesserung von Prozessen in Hinblick auf Entscheidungsmodelle, ermöglichen.

Im zweiten Teil werden Designprinzipien des serviceorientierten Architekturparadigmas verwendet, um so die system-

atische Abstraktion und konsistente Integration von Prozess- und Entscheidungsperspektiven voranzutreiben. Hierbei wird ein mehrschichtiges Referenzmodell vorgeschlagen. welches die beschriebenen Designprinzipien berücksichtigt. Dieses Referenzmodell wird anschließend anhand eines Fallbeispiels mit realen Datensätzen im Rahmen von Process Discovery diskutiert.

Im dritten und letzten Teil werden dann Adaptionsmuster für entscheidungsbasierte Prozesse präsentiert. In diesem Zuge werden ebenfalls prototypische Modellierungswerkzeuge entwickelt, welche die identifizierten Adaptionsmuster verwenden, um die Entwicklung und Erweiterung von Prozess- und Entscheidungsmodellen zu unterstützen. Die präsentierten Prototypen beinhalten Modellierungsumgebungen mit integrierten Funktionalitäten zur Konsistenzverifikation, automatischem User-Feedback sowie semiautomatischen Mechanismen zur Wiederherstellung von inter- und intramodellorientierter Konsistenz.

Zusammenfassend betrachtet überträgt diese Arbeit eine System-Modellierungsperspektive auf die Domäne der Informationssysteme mit “process-awareness”. Als Ergebnis entstehen hieraus entscheidungsbasierte Informationssysteme, sogenannte “decision-aware information systems”. Durch diesen Beitrag soll die Wichtigkeit von Entscheidungsmodellen im Rahmen der modellgetriebenen Systementwicklung hervorgehoben werden.

Resumen de Tesis Doctoral

Los sistemas de información que tienen en cuenta procesos de negocio dependen del conocimiento acerca de las decisiones que son relevantes para dichos procesos. Este conocimiento guía la ingeniería, ejecución y automatización de los procesos, e impacta múltiples perspectivas de los procesos, como el flujo de control, datos y perspectivas de recursos. Por lo tanto, una integración consistente de decisiones y procesos de negocio es imprescindible para un desarrollo y gestión sólidos de los procesos. Esta tesis extiende el soporte de modelado de procesos de negocio para incluir el conocimiento sobre las decisiones relevantes para los procesos. Así pues, esta tesis presenta avances en el campo de los sistemas de información que tienen en cuenta procesos de negocio. Las contribuciones se dividen en tres partes.

La primera parte presenta la integración de procesos y decisiones desde un punto de vista de modelado. Al detectar inconsistencias en la integración de los modelos, se proponen principios y pautas para garantizar una integración consistente al modelar procesos que tienen en cuenta el conocimiento sobre las decisiones relevantes o al rediseñarlos para que tengan en cuenta dicho conocimiento.

La segunda parte se centra en los principios de diseño desde el paradigma de la arquitectura orientada a servicios para abordar sistemáticamente la separación de las perspectivas de modelado de procesos de negocio y de decisiones, así como su integración consistente. Ilustramos esta parte mediante un registro de eventos enriquecido de la vida real que se usa para deducir los procesos y las decisiones que los influencian.

Finalmente, en la tercera parte, se presentan patrones de cambio para procesos extendidos con conocimiento sobre decisiones, así como varios prototipos de herramientas de modelado que soportan la evolución de los modelos de proceso y de decisiones a través de los patrones de cambio. Las herramientas ofrecen: un entorno de modelado con capacidades de verificación de consistencia incorporadas, retroalimentación automatizada y mecanismos automatizados guiados por el usuario para restaurar la consistencia del modelo.

En resumen, esta tesis introduce una perspectiva de modelado de decisiones para sistemas de información basados en procesos de negocio, lo que permite la gestión de procesos que tienen en cuenta las decisiones pertinentes. Al hacerlo, esta tesis pone de manifiesto la importancia de los modelos de decisión en la ingeniería de sistemas de software basados en modelos.

Sažetak Doktorske Disertacije

Savremeni informacioni sistemi se, za svoj razvoj, izvršavanje i evoluciju, oslanjaju na znanje o odlukama. Inženjering procesa, njihovo izvršavanje i automatizacija su često vođeni odlukama koje utiču na procese iz više perspektiva, kao što su kontrolni tok procesa, podaci i resursi. U skladu sa tim, dosljedno integrisanje odluka i procesa predstavlja imperativ za kvalitetan razvoj i upravljanje procesima zasnovanim na znanju. Ova disertacija prezentuje pomake u oblasti procesno-svjesnih informacionih sistema nadograđivajući procese dodatnom perspektivom odlučivanja zasnivanom na modelu. Doprinosi ove disertacije mogu se podijeliti u tri segmenta.

Prvi segment se odnosi na integraciju procesa i odluka sa stanovišta modeliranja. Uočavajući nedosljednosti u integraciji modela, disertacija predlaže principe i uputstva kako bi se osigurala dosljedna integracija prilikom modeliranja ili re-inženjeringa procesa koji uzimaju u obzir odluke.

Drugi segment primjenjuje principe dizajniranja servisno-orjentisane arhitekturne paradigme da sistemski pristupi razgraničavanju odgovornosti između procesne perspektive i perspektive odlučivanja, ali i problema njihove dosljedne integracije. U tu svrhu, predložen je slojeviti referentni dizajn u koji su inkorporirani pomenuti principi dizajniranja, a koji je ilustrovan na primjeru otkrivanja procesa koji uzimaju u obzir odluke. Ova ilustracija koristi stvarni dnevnik događaja koji je obogaćen dodatnim podacima.

Konačno, treći segment uvodi tzv. obrasce promjena za procese koji uzimaju u obzir odluke, kao i prototip alate za modeliranje koji podržavaju evoluciju procesa i modela odlučivanja pomoću navedenih obrazaca promjena. Ovi alati su sačinjeni od okruženja za modeliranje sa ugrađenim mogućnostima za provjeru dosljednosti, automatsku povratnu informaciju i automatski korisnički-navođen mehanizam za uspostavljanje dosljednosti, kako unutar samog modela tako i između različitih modela.

Ukratko, ova disertacija uvodi novu perspektivu modeliranja sistema o odlučivanju u informacione sisteme koji su svjesni procesa, transformišući ih, na taj način, u informacione sisteme koji su svjesni odluka. Čineći to, prepoznaje važnost modela odlučivanja u izradi softverskih sistema koji se temelje na modelima.

Contents

Part I

Prologue

CHAPTER 1

Outline and Contributions

"U životu treba mudro da šutiš
Al riječ ako rekneš
Neka bude teška kao svaka istina
Neka bude rečena za čovjeka"

("In life you need to wisely be silent
But if you say a word
Let it be heavy like every truth
Let it be spoken for man")

Uspavanka
— *Mehmedalija Mak Dizdar*

This chapter outlines the contributions and positions the different parts of this doctoral dissertation. This dissertation presents advances in the field of process-aware information systems engineering with a focus on decision-awareness in processes. The dissertation consists of chapters that each match to a scientific paper. Thus, this dissertation is organised as a collection of papers. Please note that the papers included in this dissertation were subject to minor modifications in response to comments raised by the Doctoral Examination Committee. Figure 1.1 provides an overview of the structure of this dissertation with the chapters divided into interconnected yet distinct parts. Part I is the Prologue which discusses the outline and the contributions of the dissertations. Parts II, III, and IV contain the core contributions, while the Part V provides the Epilogue with final remarks and conclusions.

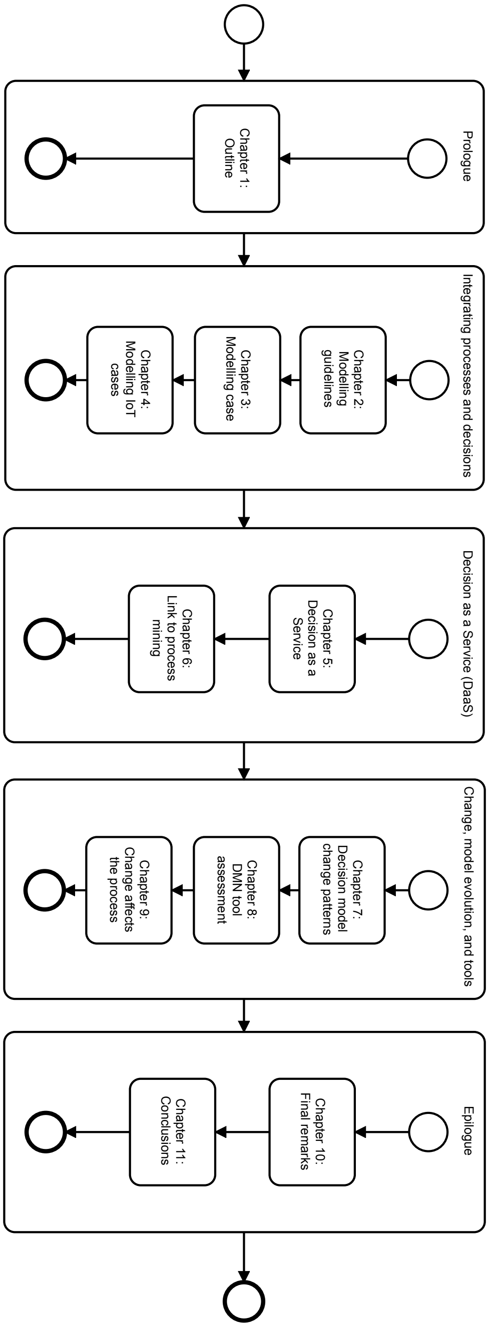

Prologue
Chapter 1: Outline
Integrating processes and decisions
Chapter 2: Modelling guidelines
Chapter 3: Modelling case
Chapter 4: Modelling IoT cases
Decision as a Service (DaaS)
Chapter 5: Decision as a Service
Chapter 6: Link to process mining
Change, model evolution, and tools
Chapter 7: Decision model change patterns
Chapter 8: DMN tool assessment
Chapter 9: Change affects the process
Epilogue
Chapter 10: Final remarks
Chapter 11: Conclusions

Figure 1.1: Overview of the dissertation.

Part I: Prologue

Chapter 1: Outline and Contributions

The overall contribution of this dissertation is the introduction of a holistic decision perspective in the development of decision-aware processes. The approach presented in this dissertation differs fundamentally from existing methods for a number of reasons that will be discussed throughout this dissertation, chief among which are the following:

- This dissertation acknowledges that decisions can occur across the entire process execution span, rather than being contained to a single decision point in the process.
- Furthermore, this dissertation recognises that decisions can be fragmented into modular parts that are distributed across the process. As such, long-distance decision dependencies are introduced into the process model.

This novel approach provides a more rigorous and holistic integration of decisions into processes while taking into consideration the separation of modelling concerns and the consistent integration between the process and decision models. The contributions provided in this dissertation have been organised into three different parts. Figure 1.1 provides an overview of the structure of this thesis. How each of these parts advance the overall contribution and how the parts are interconnected will be discussed in what follows.

Part II collects papers that present principles and guidelines for the integrated modelling of process and decision models, along with modelling cases that rely on these guidelines. The cases are taken both from industry and from literature and span different application areas. In this part, the integration of process and decision models is rather straightforward, as the integration is approached by identifying and remedying inconsistencies between a single decision model and a single process model. Here, we consider the decision model to be both correct and static, i.e., not changing over time. Given a correct and static decision model, principles for consistently integrating a single process model with the given decision model are established. Note that the integration between processes and decisions is approached from a holistic view of decisions within processes, i.e., decisions manifest themselves as modular fragments that are distributed across the entire process, thus inducing long-distance decision dependencies within the process that need to be taken into account for the consistent integration of the process and decision models.

Part III cements the consistent integration of process and decision models into the theory of the Service-Oriented Architecture (SOA) paradigm by proposing a reference design for decision-aware business processes in the form of Decisions as a Service (DaaS). Just like in Part II, here too we consider the decision model as given and static, i.e., the decision model is both correct and does not change over time. However, unlike in Part II, here we consider the integration of the decision model with processes in general, rather than a single process model. For that purpose, decision services are defined that can be used by processes to access the decision logic stored in the decision model. As such, different ways to interact with the same decision model are identified. Additionally, in Part III a link between the DaaS theory presented in this dissertation and the automated discovery of decision-aware process models from real-life enriched event logs is established, effectively illustrating that the DaaS approach manifests itself in real-life processes, i.e., that different process variants interact differently with decisions.

Finally, Part IV focuses on the consistent evolution of integrated process and decision models. Unlike in Parts II and III, here we consider dynamic decision models, i.e., decision models that change over time. More precisely, we start from consistently integrated process and decision models according to the theory specified in Part II. Next, we apply change patterns for the evolution of the decision model. The effects of the change patterns on decision model consistency are assessed, as well as their effects on the between-model consistency, i.e., the consistency between the integrated process and decision models. Subsequently, actions to remedy inconsistencies that surface as a consequence of decision model evolution are suggested. Prototype modelling environments are developed to support the modeller in the consistent evolution of decision models and decision-aware business process models.

In summary, this dissertation advances the field by:

1. Providing modelling guidelines on how to consistently integrate process and decision models. The modelling guidelines are elucidated on a handful of cases.

2. Bringing decision models to the realm of service-oriented architectures, thus cementing them in a well-established theory while linking them to the automated discovery of data-aware processes.

3. Providing an approach to consistently evolve decision models and decision-aware process models, along with proof-of-concept modelling environments with built-in consistency verification capabilities, feedback mechanisms, and automated actions that restore the consistency.

In what follows, the contribution parts of this dissertation are outlined and the different chapters of the dissertation are mapped onto papers that have either been published or that will be submitted for publication.

Part II: Integrating Processes and Decisions

This part of the dissertation is concerned about the integration of processes and decisions from a modelling perspective. It is divided into three separate yet related chapters that are briefly introduced below, along with the bibliographical references to the respective papers that the chapters represent.

Chapter 2: Augmenting Processes with Decision Intelligence

In this chapter we introduce five principles for integrated process and decision modelling (5PDM) by recognising inconsistencies in integration from the literature and from integration scenarios. The modelling principles are subsequently applied on a real-life case of a customer acceptance process in a Belgian accounting firm. The principles are applied in a step-by-step iterative fashion, effectively illustrating their application on a decision-aware process. The chapter concludes by contributing a step-wise method to ensure the consistent integration between process and decision models.

This chapter was published as follows:

Faruk Hasić, Johannes De Smedt, Jan Vanthienen. Augmenting Processes with Decision Intelligence: Principles for Integrated Modelling. `Decision Support Systems`, 107, 1-12, 2018.

A preliminary version of this paper was published in:

Faruk Hasić, Lesly Devadder, Maxim Dochez, Jonas Hanot, Johannes De Smedt, Jan Vanthienen. Challenges in Refactoring Processes to Include Decision Modelling. `Business Process Management Workshops at BPM 2017`, Barcelona (Spain), 529-541, 2018.

Chapter 3: An Illustration of 5PDM

This chapter extends the previous chapter by introducing an additional modelling case on which the integration principles defined in the previous chapter are applied. The case presented in this chapter is based on a case from the literature which was remodelled for the purpose of separating the process and decision concerns and subsequently consistently integrating the process and decision models.

This chapter presents sections 3 and 4 of the following paper:

Faruk Hasić, Johannes De Smedt, Jan Vanthienen. An illustration of Five Principles for Integrated Process and Decision Modelling (5PDM). `FEB Research Report KBI_1717 (KU Leuven)`, Leuven (Belgium), 1-8, 2017.

Chapter 4: Comparing BPMN to BPMN + DMN for IoT Process Modelling

This chapter introduces additional cases that have been modelled using the previously defined integration principles. Here, the cases are taken from Internet-of-Things (IoT) process literature and re-engineered to adhere to the principles of consistent integration. Moreover, the modelling approach from the literature is assessed against the modelling approach introduced in this part of the thesis. As such, advantages and disadvantages of the different approaches for IoT process modelling are discussed in terms of scalability, maintainability, logic reusability, and context aggregation capability.

This chapter was published as follows:

Faruk Hasić, Monique Snoeck, Estefanía Serral Asensio. Comparing BPMN to BPMN + DMN for IoT Process Modelling: A Case-Based Inquiry. `35th ACM/SIGAPP Symposium on Applied Computing (SAC)`, Brno (Czech Republic), 2020.

Part III: Decision as a Service (DaaS)

This part of the dissertation presents a service-oriented approach to decisions which we call Decision as a Service (DaaS).

Chapter 5: Decision as a Service: A Service-Oriented Architecture Approach for Decisions in Processes

The first part of this dissertation revolved around integrating a single process model with its underlying decision model. While this integration is successful from a modelling perspective, a grounded framework on the integration of processes and decisions in information systems is missing in the literature. Therefore, this chapter cements the integration of process and decision models into the well-established Separation of Concerns (SoC) and Service-Oriented Architecture (SOA) paradigms. As such, a Decision as a Service (DaaS) design that adheres to the principles of the SoC and SOA paradigms is introduced. Additionally, the manifestation of decisions as modular services is illustrated by an automated discovery of decision-aware processes from a real-life enriched event log.

This chapter was published as follows:

Faruk Hasić, Johannes De Smedt, Seppe vanden Broucke, Estefanía Serral Asensio. Decision as a Service (DaaS): A Service-Oriented Architecture Approach for Decisions in Processes. `IEEE Transactions On Services Computing`, *article in press*, 2020.

Chapter 6: Parameter Assessment of the Automated Decision Service Discovery

This chapter presents a parameter assessment of the automated discovery of decision-aware processes from a real-life enriched event log, as introduced at the end of the previous chapter. Thus, this chapter provides additional insights into the bridge between decision-aware process mining and the DaaS design pioneered in Chapter 5.

This chapter presents sections 4 and 5 of the following paper:

Faruk Hasić, Johannes De Smedt, Seppe vanden Broucke, Estefanía Serral Asensio. A Parameter Assessment of Service-Oriented Architecture Process Mining Integrating Decisions (SOAP-MInD). `FEB Research Report KBI_1914 (KU Leuven)`, Leuven (Belgium), 1-9, 2019.

Part IV: Change Patterns, Model Evolution, and Tool Support

This part of the dissertation considers the evolution of decision models and decision-aware process models through well-defined change patterns. Additionally, prototype tooling for model evolution and modelling support is presented.

Chapter 7: Decision Model Change Patterns for Dynamic System Evolution

This chapter presents a set of change patterns that can be applied on decision models in order to evolve the decision models. Applying a change pattern can result in an inconsistent decision model. To ensure model consistency, the effects of the applied change pattern on the dependent model elements need to be propagated throughout the model. As such, the referential integrity within the decision model is safeguarded. This chapter also introduces a prototype modelling environment that allows for the application of change patterns on decision models. This modelling environment allows for the application of the change patterns defined in this chapter and the environment is equipped with consistency verification capabilities providing feedback and error messages to the modeller. Additionally, the modelling environment suggests actions to remedy the detected inconsistencies. By selecting an action, the environment automatically performs the action and re-verifies the consistency of the models.

This chapter was published as follows:

Faruk Hasić, Carl Corea, Jonas Blatt, Patrick Delfmann, Estefanía Serral Asensio. Decision Model Change Patterns for Dynamic System Evolution. `Knowledge And Information Systems`, *article in press*, 2020.

A preliminary version of this paper was published in:

Faruk Hasić, Estefanía Serral Asensio. Change Patterns for Decision Model and Notation (DMN) Model Evolution. `The 18th Belgium-Netherlands Software Evolution Workshop (BENEVOL)`, Brussels (Belgium), *article in press*, 2020.

Chapter 8: A Performance Assessment of the Modelling Environment for DMN Model Evolution

This chapter builds on the modelling environment introduced in the previous chapter. More specifically, it accommodates a performance assessment of the modelling environment providing automated feedback and support for the modelling and evolution of decision models. For this purpose, random decision models, ranging from small to quite large models, were generated with induced modelling inconsistencies. Subsequently, the verification capability mechanism of the modelling environment was applied on the decision models and their run-time was recorded, effectively proving that the tool provides timely verification feedback for even the larger models.

This chapter is published as follows:

Faruk Hasić, Carl Corea, Jonas Blatt, Patrick Delfmann, Estefanía Serral Asensio. A Tool for the Verification of Decision Model and Notation (DMN) Models. `14th International Conference on Research Challenges in Information Science (RCIS), Demo Session`, Limassol (Cyprus), 2020.

Chapter 9: Consistent Evolution of Process and Decision Models

In this chapter the effects of decision model change patterns on the process model are assessed. By applying a decision model change pattern, inconsistencies between integrated process and decision models can arise. To ensure this between-model consistency, additional change patterns might be needed, either on the decision model side or the process model side. This chapter also provides a modelling environment that is capable of detecting inconsistencies between the process and decision models after a decision model change pattern is applied. The modelling environment suggests actions to remedy the inconsistencies. The modeller can select an action which is then automatically performed in order to restore the consistency between the process and decision models.

This chapter will be submitted for publication as follows:

Faruk Hasić, Carl Corea, Jonas Blatt, Patrick Delfmann, Estefanía Serral Asensio. Consistent Evolution of Integrated Process and Decision Models. *To be submitted for publication*, 2020.

Part V: Epilogue

This final part concludes this doctoral dissertation in two closing chapters.

Chapter 10: Final Remarks

In Chapter 10 we provide a few final remarks, clarifications, and assumptions requested by the Doctoral Examination Committee with regard to the contributions detailed in Parts II - IV.

Chapter 11: Conclusions

This final chapter wraps up this doctoral dissertation. Here, we summarise the contributions and limitations, and we shed a light on future research opportunities that build on the findings of this dissertation.

Part II

Integrating Processes and Decisions

CHAPTER 2

Augmenting Processes with Decision Intelligence

"What is the cause of the unification? In all things which have a plurality of parts, and which are not a total aggregate but a whole of some sort distinct from the parts, there is some cause."

Metaphysics, Book 8
— *Aristotle*

This chapter was published as follows:

Faruk Hasić, Johannes De Smedt, Jan Vanthienen. Augmenting Processes with Decision Intelligence: Principles for Integrated Modelling. `Decision Support Systems`, 107, 1-12, 2018.

A preliminary version of this paper was published in:

Faruk Hasić, Lesly Devadder, Maxim Dochez, Jonas Hanot, Johannes De Smedt, Jan Vanthienen. Challenges in Refactoring Processes to Include Decision Modelling. `Business Process Management Workshops at BPM 2017`, Barcelona (Spain), 529-541, 2018.

Abstract. Until recently decisions were mostly modelled within the process. Such an approach was shown to impair the maintainability, scalability, and flexibility of both processes and decisions. Lately, literature is moving towards a separation of concerns between the process and decision model. Most notably, the introduction of the Decision Model and Notation (DMN) standard provides a suitable solution for filling the void of decision representation. This raises the question whether decisions and processes can easily be separated and consistently integrated. We introduce an integrated way of modelling the process, while providing a decision model which encompasses the process in its entirety, rather than focusing on local decision points only. Specifically, this chapter contributes formal definitions for decision models and for the integration of processes and decisions. Additionally, inconsistencies between process and decision models are identified and we remedy those inconsistencies by establishing **Five Principles** for integrated **P**rocess and **D**ecision **M**odelling (**5PDM**). The principles are subsequently illustrated and validated on a case of a Belgian accounting company.

2.1 Introduction

The prevalence of new works on decision modeling and mining, as witnessed by the vast amount of new works on Decision Model and Notation [61, 64, 78, 106, 108], shows an increasing interest in documenting, modelling, and analysing the decision dimension of processes. DMN has two levels that are to be used in conjunction. Firstly, there is 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. Secondly, there is the decision logic level, which presents ways to specify the underlying decision logic. Usually, the decision logic is specified in decision table form. An example of a DRD is given in Figure 2.1. DMN is designed as a declarative decision language. As a result DMN provides no decision resolution mechanism, as this is left to the invoking context (e.g. a process). The same holds for the processing and storage of outputs and intermediate results. Besides DMN, also the Product Data Model (PDM) [136] is a well-known language to capture the dependencies that exist between decisions and their input in workflows. DMN, however, is more driven by the decision and its rationale compared to PDM, which rather focuses on the data and its impact on the workflow.

Organisations use Business Process Management (BPM) and Decision Management (DM) to analyse, and improve their processes. The new DMN standard has the clear intention to be used in conjunction with Business Process Modeling and Notation (BPMN) [16, 37, 50, 61, 137]. Since the introduction of DMN, the general consensus is to model decisions outside processes. BPM is moving towards this *separation of concerns* paradigm [52] by externalising the decisions from the process flow.

The contribution of this chapter is fourfold: (1) a formal definition of decision models and their relation to process models is established; (2) a list of inconsistencies between process and decision models is provided based on existing literature and on the formal definitions formulated in this chapter; (3) a set of modelling guidelines is instituted to remedy the inconsistencies between process and decision models. The guidelines are contributed in the form of **Five P**rinciples for integrated **P**rocess and **D**ecision

Modelling (**5PDM**), in analogy with [94]; (4) the proposed modelling principles are applied and tested on a real life industry case.

This chapter is structured as follows. In Section 2.2 the design science approach used in this chapter is explained, while Section 2.3 handles the necessities for integrated modelling and decision modelling. In Section 2.4 a formalisation of the DMN standard and related constructs is provided which will serve as the basis for the approach of integrated modelling. Section 2.5 outlines challenges of integration by providing scenarios containing inconsistency concerns, followed by Section 2.6 which extracts principles for integrated process and decision modelling from the previous sections. In Section 2.7, the modelling principles are illustrated on a case from industry, and in Section 2.8 a systematic approach to mitigate inconsistencies is provided. Finally, Section 2.9 discusses the contributions and future work.

2.2 Methodology

This chapter follows a design science approach [73], structured along three different cycles to obtain an artifact, being the **5PDM**. First of all, the *application domain* and *population* was delineated as practitioners who develop models for integrating decisions into processes for process-aware information systems during the relevance cycle. Next, we have identified the *problem* of *inconsistent* use of *decisions* within *processes* and hence the issues that arise regarding maintainability, scalability, flexibility, understandability and reusability of decisions and processes in Sections 2.1 and 2.3. We have argued that these are the *relevant* issues tackled when separating concerns in modelling endeavours through the use of the separation of concerns and Service-Oriented Architecture paradigms in Sections 2.4 and 2.5. Based on the previous work of the authors, a literature review, and insights from industry (i.e. the case study environment), it was noted that there are *no suitable guidelines*, and that from previously produced models in research *no streamlined approach* was suggested. Next, an initial set of guidelines, i.e. the proposed *solution artefact*, were built in Section 2.6, according to examples from practice and research. They were *validated by practitioners*, as illustrated in Section 2.7, and previous work [61], during the design cycle. Finally, this chapter aims at *formalising the procedure* to adhere to the guidelines in Section 2.8 and bringing them to the body of literature on decision and process modelling. Note that these cycles work like cogs, and the *relevance cycle* was influenced both by insights from literature, as well as practice and design iterations, while the *rigor cycle* produced initial findings which were reflected in the design.

2.3 Why Integrated Decision and Process Modelling?

This section provides a motivation and related work for separating and integrating process and decision models. Additionally, we provide a running example that will be used throughout this chapter.

2.3.1 Motivation and related work

In the trend towards integration several situations can be identified. Basic solutions see processes represented using only BPMN, or decisions using only DMN. This approach works only in the most straightforward cases, where no decisions are made during the process, or where only the result of a single decision is needed respectively. Slightly more evolved situations see a complete decision model represented by a single activity in a business process. This approach will only be valid for straightforward processes and decisions. Decisions are often emulated using intricate control flows, which can result in cascading gateways. These hidden decisions must be identified in the process. After identifying and modeling these decisions the resulting model must be integrated consistently with the process model. This insufficient separation of concerns results in maintainability issues [61, 62, 63, 115]. In more complex processes several decisions might influence both the flow and the result. Representing these decisions and invoking them correctly in the process is crucial for a proper understanding of the process. However, these more convoluted situations have encountered little consideration in literature.

Research has already dealt with data-aware processes and process consistency regarding data management [127]. Extensions regarding data-awareness in process modelling have been proposed as well [28]. In [109] an ontology-based knowledge-intensive approach is suggested, while [95] proposes an enhancement of declarative process models with DMN logic. Furthermore, works concerning data-aware/coloured Petri Nets are available as well, offering a formally sound approach to data and process integration [120]. However, merely focusing on data fragments is not sufficient to incorporate decision-awareness into processes, which DMN aims to achieve. Furthermore, it has been illustrated how the Decision Model [138], a decision representation similar to DMN, can be used within business process models as well [1] to offload the control flow from embedded decisions. The findings of this chapter are also compatible with the Decision Model after a straightforward conversion of its decision and input blocks into DRD constructs.

The decision modelling approaches present in literature often breach the separation of concerns between control and data flow, resulting in spaghetti-like processes, thus negatively influencing maintenance, flexibility, scalability and reusability [61, 63, 65, 79, 115, 132, 137, 146, 151]. They do this by hard-coding and fixing the decisions in processes. Consequently, splits and joins in processes are misused to represent typical decision artifacts such as decision tables. Recently, more attention was given to the separation of processes and decision logic, as such an approach is supported by the DMN standard [106] that can be used in conjunction with BPMN [61, 63, 105]. Decoupling decisions and processes to stimulate flexibility, maintenance, and reusability, yet integrating decision and process models is therefore of paramount importance [61, 63, 84].

The separation of concerns has enjoyed plenty of attention, mainly in the domain of software modelling and design [52], but recently it has become an evident trend in BPM as well. This has moved decision management towards the paradigm of Service-Oriented Architecture (SOA), by representing decisions as externalised services. In research several conceptual decision service platforms [22, 151] and ontologies [89] have been proposed. Separation of concerns and SOA offer firm motivation for keeping multi-perspective modelling tasks isolated and founded on a basis which can be used to ensure consistency. The integrated modelling and externalisation was already considered in terms of business rules [51, 141]. With DMN, externalisation of decisions has become a possibility, since decisions can be encapsulated in separate decision models. These decisions are modelled separately from other concerns, such as processes, and they are implemented as a service which we call *Decision as a Service (DaaS)*. Other information systems, e.g. process-aware information systems, can invoke the decision services from the separate decision layer on demand, i.e. *Decision on Demand (DoD)*. Consequently, a decision model can be invoked and used as a service, adhering to the SOA paradigm and benefiting maintainability, scalability, flexibility, and reusability [16, 51, 52, 61, 63, 79, 84, 96, 132, 137, 151]. This emphasises the necessity for a separate, yet integrated modelling of decisions and processes.

2.3.2 Running example

In this chapter the integration of decision and process modelling will be elucidated through a case study in a Belgian accounting firm. By law, Belgian accounting firms are obligated to provide a decision model to the public authorities on which they base their decision of accepting or rejecting customers. Figure 2.1 depicts the decision model for customer acceptance at the firm. `Customer Acceptance` is decided based on the customer's `Risk Level`, which on its turn depends on a `Financial Position Check` and a `Background Check` of the customer. The decision logic is externalised to this model and a complementary process model is provided in Figure 2.2.

This process model will have to comply with the decision model in order to correctly fulfill the customer acceptance process, i.e. the process model must be modelled consistently with the decision model. However, Figure 2.2 contains plenty inconsistencies, as will be discussed in the following sections.

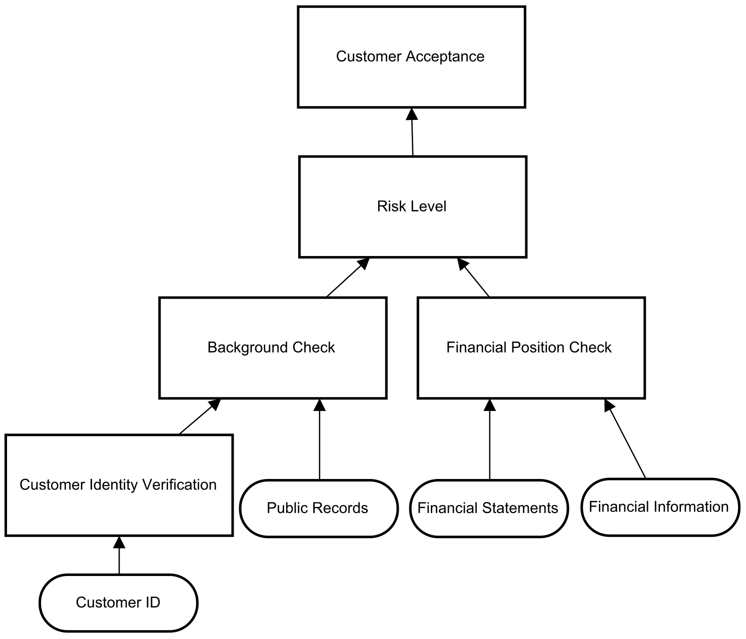


Figure 2.1: Decision model for customer acceptance at a Belgian accounting firm.

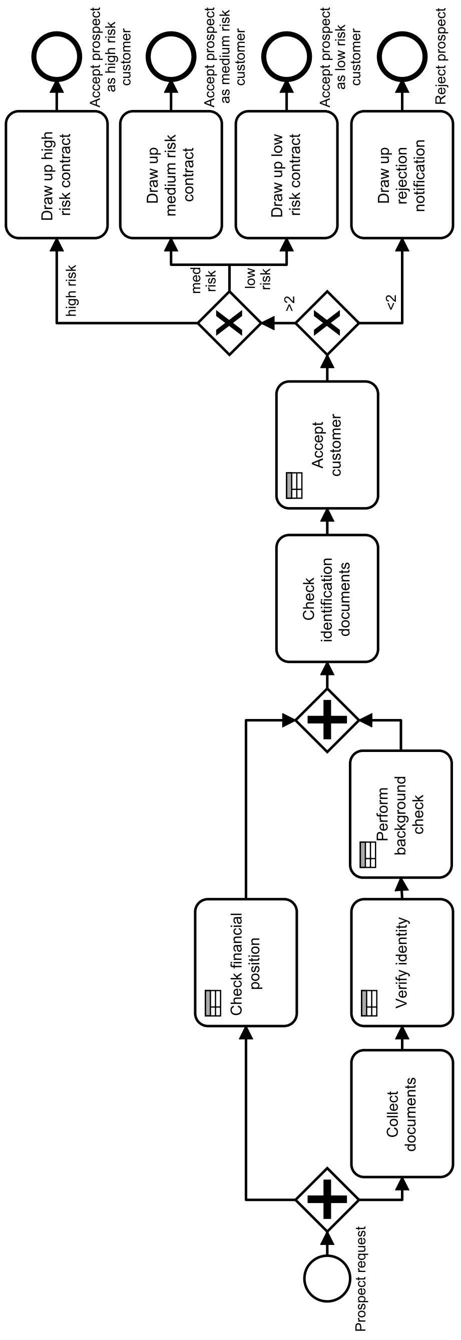

Prospect request
Collect documents
Verify identity
Perform background check
Check financial position
Check identification documents
Accept customer
>2
<2
high risk
med risk
low risk
Draw up high risk contract
Draw up medium risk contract
Draw up low risk contract
Draw up rejection notification
Accept prospect as high risk customer
Accept prospect as medium risk customer
Accept prospect as low risk customer
Reject prospect

Figure 2.2: Process model for customer acceptance at a Belgian accounting firm.

2.4 Formal Definitions

In this section, a formal basis is given for DMN constructs and for the connection between decisions and processes, which is key for the integration.

2.4.1 Basic DMN constructs

We formalise DMN to aid us in the consistent integration of processes and decisions. We adopt the definition of *decisions* and *decision requirement diagrams* from [84, 122] and expand them to include subdecisions, interfaces, and invocability.

Definition 2.1. A decision requirement diagram DRD is a tuple (D_{dm}, ID, IR) consisting of a finite non-empty set of decision nodes D_{dm}, a finite non-empty set of input data nodes ID, and a finite non-empty set of directed edges IR representing the information requirements such that $IR \subseteq (D_{dm} \cup ID) \times D_{dm}$, and $(D_{dm} \cup ID, IR)$ is a directed acyclic graph (DAG).

The DMN specification allows a DRD to be an incomplete or partial representation of the decision requirements in a decision model. The complete set of requirements R_{DM} is derived from the set of all DRDs. The information contained in this set can be combined into a single DRD representing the entire decision requirements level, i.e. the decision requirement graph (DRG). We extend the notion of a DRG, in such a way that a DRG is a DRD which is self-contained as explained in Definition 2.2.

Definition 2.2. A decision requirement diagram $DRD \in R_{DM}$ is a decision requirement graph DRG if and only if for every decision in the diagram all its modeled requirements, present in at least one diagram in R_{DM}, are also represented in the diagram.

According to DMN a decision is the logic used to determine an output from a given input. In BPMN a decision is an activity, i.e. the act of using the decision logic. Another common meaning is that a decision is the actual result, which we call the output of a decision. We define a decision using its essential elements.

Definition 2.3. A decision $d \in D_{dm}$ is a tuple (I_d, O_d, L), where $I \subseteq ID$ is a set of input symbols, O a set of output symbols and L the decision logic defining the relation between symbols in I_d and symbols in O_d.

In case of decision tables, a commonly used reasoning construct in decision models, I_d and O_d contain the names of the input, and output elements, respectively, and L is the table itself, i.e. the set of decision rules present in the table. Note that, since a DRD is a DAG, $I_d \cap O_d = \emptyset$. In DRDs these decisions d_i are represented by the decision nodes $D_i \in D_{dm}$. We will use D to refer to both a decision and its representing node in a DRD. From the definition of DRGs we can derive an important property of decisions. From Definition 2.2 we know that a DRG contains exactly all information requirements of its decisions. Thus there can only exist one DRG with D as its single top-level decision. We use DRG_D to denote this DRG.

To identify incorrect uses of decisions in process models it is important to know their structure. Decisions are often structured hierarchically.

Definition 2.4. A decision D' is a subdecision of decision D if and only if it is part of DRG_D, but not D itself.

This order of decisions and subdecisions can be defined by using the property that DRDs are directed acyclic graphs, from Definition 2.1. From this property we know that each DRD has a topological order. The concept of topological orders is closely related to partial orders resulting in Property 1.

Property 1. *The topological order of a DRD induces a partial order $\leq$ on the decisions contained in the DRD.*

When integrating decisions with processes, reducing the coupling between both is important for flexibility and maintainability, as stated by the Service-Oriented paradigm. In this paradigm, decisions are viewed as an external service which exists as a unit decoupled from the process that can be invoked by the process. In [63, 96], the advantages of decision services are elaborated on, showing that the Service-Oriented paradigm enables flexibility, maintainability and scalability. This is achieved by making abstraction of decisions in the process and only connecting the process to the decisions through an interface of the decision service. In order to define the interface, Definition 2.5 first defines the input requirement set.

Definition 2.5. The decision input requirement set $dirs_D$ of a decision D is the set of all sets of input data which are sufficient to invoke D. $dirs_D$ contains sets of input data directly or indirectly required by D. The largest set in $dirs_D$ is the set of all input data nodes for which there exists a path to D in DRG_D. The smallest set in $dirs_D$ is D's input set I_D. $dirs_D$ is constructed inductively by the following rule:

$I_D \in dirs_D$ and for all $s \in dirs_D$ if there is an $i \in s$ such that $i \in O_{D'}$ for some D' in DRG_D, then $s \setminus \{i\} \cup I'_D \in dirs_D$.

A decision's interface is the combination of its input requirement set and its output set. Thus, decision interfaces can be defined as in Definition 2.6.

Definition 2.6. The interface IF_D of a decision D is a tuple $(dirs_D, O_D)$, where $dirs_D$ is the input requirement set and O_D the output set of D.

In DMN, decisions are constrained to have no side-effects so they comply with the principles of a service. As such each decision, with its associated interface, can be seen as a decision service. Consequently only the information available in a decision's interface should be used in the process. Each decision in a DRD has its own output set and these sets should be disjoint. The outputs of subdecisions are identified as *intermediate results*. An output O is an intermediate result of decision D if and only if $O \notin O_D$ and there exists a subdecision D' of D for which $O \in O_{D'}$.

Executing a decision in DMN is referred to as *invoking* the decision. Using the definition of a decision's interface it becomes possible to define when a decision can be invoked. Generally a decision can only be made if all required inputs are available. This is especially important when decisions are invoked in a process. Definition 2.7 determines the invocability of a decision.

Definition 2.7. A decision D is invocable from a set of data elements S if there exists an $s \in dirs_D$ such that $s \subseteq S$. Given at least the values of all data elements in one of the sets in $dirs_D$ the output of D can be determined.

If a decision is invocable from a set of input data nodes in the DRD, so are its subdecisions, as stated in Theorem 1:

Theorem 1. *If a decision D is invocable from a set of input data node objects S, then so are all of its subdecisions.*

Proof for Theorem 1. *Assume D' is a subdecision of D. Since there is a path from D' to D in DRG_D, there is also a path from each of the input requirements of D' to D. Thus, $dirs_{D'} \subseteq dirs_D \subseteq S$.*

We have formalised relevant constructs in the decision model. However, in order to discuss model integration, the decision model must be correlated with the process model. We formalise that connection in what follows.

2.4.2 The key to integration: decision activities and intermediate results

In this subsection, we propose a typology for different activities used for making decisions in processes. By doing so, we will link the decision model to the process model requiring the decisions. Decisions do not surface solely as the driver of control flow. Rather, they both encompass the routing of cases, i.e., because of decision outcomes that steer toward a certain activity tailored towards supporting its output, and the changes in the data layer of the process as well. For a consistent integration, distinguishing between decision making activities and intermediate results is of paramount importance, especially in the case of separation of concerns, where the decision model is externalised and holistically integrated with the process model. This categorisation is imperative for the identification of types of activities that are representatives of the *decision* model in the *process* model:

Definition 2.8. The input and output data variables of process activities in A are defined as follows:
$I : A \rightarrow V$, function assigning activities that deliver input for a variable,

$O : A \rightarrow V$, function assigning activities that deliver output for a variable.

This enables the construction of the following activity types:

1. **Operational activities ((no) inputs, no outputs):** do not have any influence on the process' decision dimension and only act as a performer of an action that is tied to that specific place in the control flow. They might serve as the conclusion of a decision. They are provided with the decision inputs needed, which are not used further in the process, $A_o = \{a \in A \mid O(a) = \emptyset, \}$.

2. **Administrative activities (no inputs, outputs):** they introduce decision inputs into the process, $A_a = \{a \in A \mid I(a) = \emptyset \wedge O(a) \neq \emptyset\}$.

3. **Decision activities (inputs, outputs):** serve a decision purpose by transforming inputs into an outcome, $A_d = \{a \in A \mid I(v) \neq \emptyset \wedge O(v) \neq \emptyset\}$.

It holds that $A_a \cup A_o \cup A_d = A$. Typically, the decision points that are used for decision mining in processes are of the decision activity type, but tailored towards deciding which activity should be performed next based on the event labels, instead of encompassing the process in its entirety.

We can now make the connection with decisions and process models:

Definition 2.9. A decision in a business process can be defined as follows:

A decision in a process model, $d^a \in D_{dm}$ is a tuple $(I_{d^a}, O_{d^a}, L_{d^a})$, where $a \subseteq A_d$, $I_{d^a} \subseteq I(a)$, $O_{d^a} \subseteq O(a)$ and $L_{d^a} \subseteq L$.

Now that we have defined the connection between decision activities in the process and the decision in the decision model, we can also define process-decision model consistency. Given a decision model, the process model should ensure that the decisions it invokes are invocable at that point in time, and that the decision results can only be used by the process if they have been invoked explicitly by said process. Hence, keeping in mind the previous definitions, we can define consistency for a process-decision model:

Definition 2.10. A process model is consistent with a decision model if and only if the following two conditions hold:

1. No intermediate results of non-invoked subdecisions are used.

2. Each (sub)decision invoked in the process, must be guaranteed to be invocable at that stage of the process.

2.5 Integration Scenarios and Inconsistencies

In this section we shortly describe possible integration scenarios and subsequently extrapolate inconsistencies that might occur in those scenarios.

2.5.1 Integration scenarios

Outlines of possible process-decision integration scenarios are provided by [61, 84]. We refer to those papers for a full description of possible integration scenarios. Two extreme scenarios occur when there is only one model and hence no need for integration: a scenario describing a simple process without decisions, and a scenario with decisions where no actual process is needed. A third scenario with only one model is possible: both decisions and processes are present, but they are intertwined within the same model, as decisions are hard-coded within the process. This scenario clearly breaches the separation of concerns paradigm. A fourth scenario treats decisions as local concerns, as part of the decision logic pertaining to XOR-gates within the process is separately encapsulated in a decision model. A more challenging scenario exists when, instead of dealing with local decisions, interrelated decisions span over multiple activities of the process. These decisions will influence the process in multiple ways, not only in terms of control flow at gateways. They will shape the flow of the process, the outcome of the process and the process modelling itself, as will be illustrated in the coming sections. The current scenario establishes long-distance dependencies between activities, data, control flow, and decisions in the process model, enabling a decision model to span over multiple activities instead of being contained to a single decision point in the process. Data management to liaise the data generated by activities that feed into decisions will be paramount for the integration of such process and decision models.

2.5.2 List of inconsistencies

In this subsection, possible inconsistencies that might arise between the process model and the decision model are described, as the goal is to identify potential inconsistencies and subsequently to alter the process to restore consistency. Furthermore, we formally define the (in)consistencies based on the formal definitions from the previous sections. All the inconsistencies are also directly linked with the relevant decisions and properties, which all heavily rely on the integration definition, i.e., Definition 2.10.

***Inconsistency 1 (I1)* - exclusion of decision outcomes:** Not all outcomes from the decisions are included in the process model. Decisions can (re)direct the flow of the process. In an integrated process-decision model, all outcomes of the decision should be represented in the control flow if that decision redirects the process. Modelling all possible decision outcomes in the process is vital for a correct conclusion of the process.

Formally, if a decision D with an output set O_D in the decision model DM leads to a change in control flow in the process, then all elements of O_D should be present in the control flow resulting from the decision activity A_D which links D from DM to the process, i.e., in the state space of the process, the occurrences of $o \in O_D$ lie between the occurrences of A_D and the accepting states. This builds on Definition 2.9.

***Inconsistency 2 (I2)* - inclusion of decision logic in the process:** An inappropriate way to model parts of the decision logic is to embed decision logic in gateways. In cases where a process contains decision logic, the process is incapable of accommodating to changes in the underlying decision model. When changes occur, the process itself needs to be adapted. This occurs when the separation of concerns is not adopted strictly and thus the decision logic is not separated and encapsulated in an independent decision model. Hence, **I2** does not allow for evolution of both models disjointedly.

More precisely, the decision logic L of a decision D should not be part of the process. Rather, L belongs to a decision $D \in D_{dm}$, where D_{dm} is the finite non-empty set of decision nodes belonging to the DRD. Hence, the L is encapsulated in the decision model DM. The process can invoke D through its interface IF_D, which was defined as a tuple $(dirs_D, O_D)$. Through IF_D, the process provides the input requirement set $dirs_D$ needed for the enactment of D, and again through the interface, the decision model returns the output set of D, i.e. O_D. Hence, the process accesses the decision model through an interface and is agnostic of the underlying decision logic. This builds on Definitions 2.5 and 2.6.

***Inconsistency 3 (I3)* - exclusion of intermediate results:** Inconsistencies arise when the process uses the outcomes of subdecisions that are not explicitly modelled in the process. These intermediate results may be needed in the process to serve as inputs for operational activities or as inputs to higher-level decisions. Hence, a process model that is consistent with the decision model should ensure that all the subdecisions that contain intermediate results that are used in subsequent activities are explicitly invoked.

Explicitly, if the process uses the intermediate result $O_{D'}$ of a subdecision D' of higher level decision D, then D' must be represented by a decision activity $A_{D'}$ in the process. As such the process can invoke D' through the subdecision's interface $IF_{D'}$ by providing the necessary input requirements from $dirs_{D'}$, after which $IF_{D'}$ will provide $O_{D'}$ to the process. This builds on Definitions 2.6 and 2.10.

***Inconsistency 4 (I4)* - inclusion of process-unrelated subdecisions:** Opposite to **I3**, more decisions than necessary can be included in the process. This occurs when decisions which do not contain relevant intermediate results for the process are modelled within the process. In this case the process becomes unclear and overly complex. Along with that, by modelling every subdecision in the process, the decision enactment or execution steps become fixed. This contradicts the declarative nature of decisions and reduces the flexibility provided by the decision model.

Specifically, if a subdecision D' with an output set $O_{D'}$ in the decision model DM does not lead to a change in control flow in the process, and if the process does not use intermediate result $O_{D'}$ of D', then no decision activity $A_{D'}$ representing subdecision D' should be modelled in the process. This builds on Definitions 2.7 and 2.9.

***Inconsistency 5 (I5)* - unsound ordering of decision hierarchy:** This occurs when the order of decision activities in the process model is contradictory to the hierarchy of decisions in the decision model. Consequently, the process cannot function correctly, as decisions are forced to enact without the prerequisite enactment of necessary subdecisions. This order of decisions and subdecisions introduces a partial order as shown in Property 1.

Hence, for two decisions D_1 and D_2 we say $D_2 \leq D_1$ if and only if there is a directed path from D_2 to D_1, i.e. D_2 is a subdecision of D_1. Since decisions are declarative, this partial order does not dictate an execution order, but rather a requirement order. Using this order induced by Property 1 and the result from Theorem 1 we know that if a decision D is invoked in the process any decision D' for which $D' \leq D$ will be invocable when placed directly in front of D.

***Inconsistency 6 (I6)* - exclusion of subdecisions affecting control flow:** Depending on the outcome of certain subdecisions the control flow of the process may be diverted to include additional activities, to generate exceptions or even to lead to process termination. Excluding these subdecisions that have an influence on the control flow of the process leads to process-decision inconsistency. This inconsistency is closely related to **I3**: while **I3** focuses on the exclusion of generated data by certain subdecisions, this inconsistency focuses on the change of control flow.

Explicitly, if a subdecision D' with a decision output set $O_{D'}$ in the decision model DM leads to a change in control flow in the process, then D' must be represented by a decision activity $A_{D'}$ in the process. Additionally, all elements of $O_{D'}$ should be reflected in the control flow following $A_{D'}$ which links the subdecision D' from the decision model DM to the process. This builds on Definitions 2.7 and 2.9.

***Inconsistency 7 (I7)* - absence of input data:** Decision activities require prerequisites to function correctly. These prerequisites can be the outcome of certain subdecisions, as illustrated in **I3**, but also take the form of for instance user-generated input data. The inconsistency in this case occurs when the required input data is not available in a process when a certain decision task needs to be executed.

Formally, if the process contains a decision activity A_D referring to a decision D in the decision model DM, then the process must make sure that $dirs_D$, i.e. the required input set for the decision D, is available within the process at the time decision activity A_D is executed. Only then can the process invoke decision D through its interface IF_D. This builds on Definitions 2.5 and 2.10.

2.6 Principles for Consistent Integration

In this section we provide a set of principles for integrated process and decision modelling. The principles are derived based on the integration scenarios and the formalisation from the previous sections. The principles state what should be included in a process model and what should be excluded from a process model which is linked to a corresponding decision model. **Five P**rinciples for integrated **P**rocess and **D**ecision **M**odelling (**5PDM**) are derived to support consistency between the two models:

P1. Model all necessary decision output flows. If after enacting the decision, no output flow is dedicated to the decision outcomes, the process will prove to be inconsistent. Namely, if a decision outcome that is not modelled in the control flow of the process occurs after the decision enactment, then the process cannot proceed properly.

P2. Do not include decision logic in the process model. Otherwise, maintainability, flexibility and scalability of the process might be impaired. Rather, the underlying decision logic should be externalised, encapsulated in the decision model and invoked as a service by the process.

P3. Model all subdecisions whose intermediate results are used by or are relevant for the process as decision activities in the process (**P3.1**). Subdecisions can only be invoked explicitly if they are represented in the process model. Using intermediate results of subdecisions that are not represented in the process leads to data inconsistency between the process and decision model. If a certain subdecision directly impacts the process control flow, the decision should be explicitly represented in the process model

Table 2.1: 5PDM: five principles for integrated process and decision modelling.

Principles for integrated process-decision modelling (5PDM)
P1: `Include` *all* necessary *decision outcomes* in the process control flow
P2: `Exclude` *decision logic* and cascading XOR-splits from the process
P3: `Include` only *subdecisions* that directly *influence* the process
P3.1: `Include` *subdecisions* whose *results* are used in the process
P3.2: `Include` *subdecisions* that *affect* the process *control flow*
P3.3: `Exclude` *subdecisions* that are or *irrelevant* to the process
P4: `Include` *decision hierarchy* in decision activity modelling
P5: `Include` *input data* and *intermediate results* for decision enactment

by a corresponding decision activity (**P3.2**). The decision might steer the control flow of the process towards additional activities, exception handling or even process termination. Excluding such decisions from the process leads to inconsistency. However, do not include more decision activities than necessary. Only the top-level decision and subdecisions relevant for the process enactment in terms of control flow, intermediate results and data management should be represented in the process itself. All other process-irrelevant subdecisions should not be modelled explicitly in the process (**P3.3**). Furthermore, modelling all subdecisions violates the declarative nature of decision modelling and reduces the flexibility provided by the decision model.

P4. Place all relevant decision activities in the correct order within the process. This is paramount for the correct enactment of the process and the underlying decisions, since intermediate results of subdecisions are often needed later in the process to enact higher level decisions. Modelling the subdecision before the higher level decision in the process is therefore vital for a correct management of intermediate results and data and hence for a proper decision and process enactment. Disregarding the decision hierarchy results in an inconsistent process-decision model.

P5. Model all data objects and intermediate results necessary for a correct process and decision enactment. The decision model depicts the hierarchy of decisions and hence the inputs and intermediate results that are necessary for enacting the decisions that are relevant for the process. If not all required data are represented in the process model, the decision activities requiring that data will not be executed properly and ensuring a sound process enactment becomes a difficulty. Thus, the process must facilitate a correct data management to be able to invoke a decision through its interface.

A short overview of the **5PDM** principles is provided in Table 2.1.

2.7 How to integrate decision and process models

In this section, we will address the inconsistencies in the running example of Figure 2.2 and rework it according to the 5PDM. We have also developed a second example, however, due to page constraints that example is not incorporated in this chapter. Rather, the example is available online in Section 3 of a *technical report* of our home institution [60][1].

2.7.1 Inclusion of all decision outcomes in the control flow

A first concern with the process model provided in Figure 2.2 is that not all possible outcomes of the `Accept Customer` decision activity are represented in the control flow of the process model. The XOR-split that follows the `Accept Customer` decision activity offers flows for cases where the score is larger than 2 or smaller than 2. However, for a score equal to 2 there is no corresponding path in the process model. This situation corresponds to **I1**.

To remedy this inconsistency, **P1** will be used. The process model should include a flow that supports the decision outcome of a score equalling 2. A solution is presented in Figure 2.3, where an additional flow exits the XOR-gate and guides the indecisive cases with score equal to 2 towards the executive meeting where the customer acceptance will be resolved.

[1]This example will be presented in Chapter 3 of this thesis.

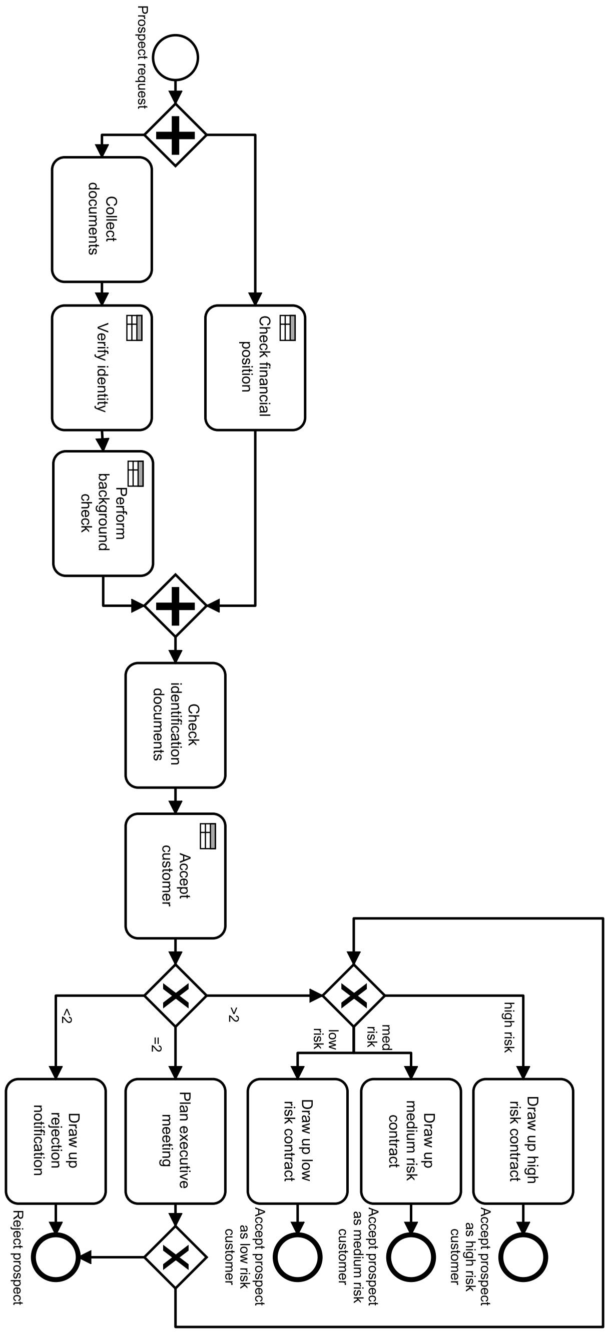

Prospect request
Collect documents
Verify identity
Check financial position
Perform background check
Check identification documents
Accept customer
<2
=2
>2
low risk
med risk
high risk
Draw up rejection notification
Plan executive meeting
Draw up low risk contract
Draw up medium risk contract
Draw up high risk contract
Reject prospect
Accept prospect as low risk customer
Accept prospect as medium risk customer
Accept prospect as high risk customer

Figure 2.3: Iteration 1.

2.7.2 Exclusion of decision logic from the process model

Figure 2.3 includes all possible decision outcomes at the XOR decision point after `Accept Customer`. However, part of the decision logic is included in the process model, as the business rule that determines the outcome of decision activity `Accept Customer` has been hard-coded in the control flow of the XOR decision point. If the resulting score is smaller than 2, the customer will be rejected; if the score is larger than 2, the customer will eventually be accepted; and if the score equals 2, the executive meeting will address the customer acceptance issue. Suppose that along the way, this business rule for customer acceptance changes and that a score higher than 4 leads to customer acceptance; lower than 4 to customer rejection; and equal to 4 to the executive meeting. This business rule can be easily changed in the decision model. However, once the rule is changed, the process model does not longer consistently comply with the decision model. Hence, in this case, changes in the process model are necessary as well to achieve consistency. This corresponds with **I2** and should be remedied by **P2**. Better is not to include decision logic in the process model and to simply model the outcome of the decision, as done in Figure 2.4. Instead of hard-coding the scores in the control flow, only the decision outcome of *accept, pending, reject* is modelled in the control flow. The actual decision logic is encapsulated in the decision model, improving the agility and maintainability of the process model.

Figure 2.3 contains different constructs containing decision logic. After a customer gets accepted, the XOR-gateway following the acceptance resembles a decision tree, where a distinction is made between low risk, medium risk and high risk customers. These are outcomes of the subdecision `Risk Level` in the decision model in Figure 2.1. They all lead to similar activities of drawing up a contract. Hence, there is no need to model outcomes of a subdecision in the control flow, as this is another instance of including decision or data flows in the process model. The outcome of the `Risk Level` subdecision is determined in the decision model. Since the top-level decision `Customer Acceptance` is invoked by decision activity `Accept Customer`, the subdecision `Risk Level` is also implicitly invoked and there is no need to additionally model that part of the decision in the control flow. This is again a typical instance of **I2**, as control flows are often misused to represent decision logic. This issue is treated according to **P2** in Figure 2.4, where the redundant gateway is excluded from the process model.

2.7.3 Inclusion of subdecisions directly influencing the process

I3 focuses on the use of intermediate decision results in the process model. In Figure 2.4 the `Draw up contract` activity prepares a contract for an accepted customer based on the customer's *Risk assessment file*, an intermediate result of the `Risk Level` subdecision in the decision model in Figure 2.1. However, the process model in Figure 2.4 does not recognise this intermediate result. Hence, drawing up the contract will not be possible since the *Risk assessment file* is missing. In order to include this file in the process model, the subdecision `Risk Level` that produces it should be modelled as a decision activity in the process. Figure 2.5 includes the necessary subdecision and intermediate result in accordance with **P3.1**.

Additionally, in Figure 2.4 the process remains the same regardless the validity of the identity verification executed by activity `Check Identification Documents`. In order to make the decision more reliable, the process could decide to only proceed when the identity is valid. The decision model in Figure 2.1 provides a subdecision `Customer Identity Verification`. Instead of using the generic activity `Check Identification Documents` as in Figure 2.4, it is preferential to use a decision activity referring to the subdecision `Customer Identity Verification` in the decision model. Depending on the outcome of this subdecision, i.e. whether the identity documents are valid or not, the control flow of the process can be diverted to include additional activities to ensure that a valid identity is provided before the process can continue. This is achieved in Figure 2.5 by replacing the generic activity `Check Identification Documents` by the decision activity `Verify Identity`. If the documents are deemed valid, the process can continue; if deemed invalid, an

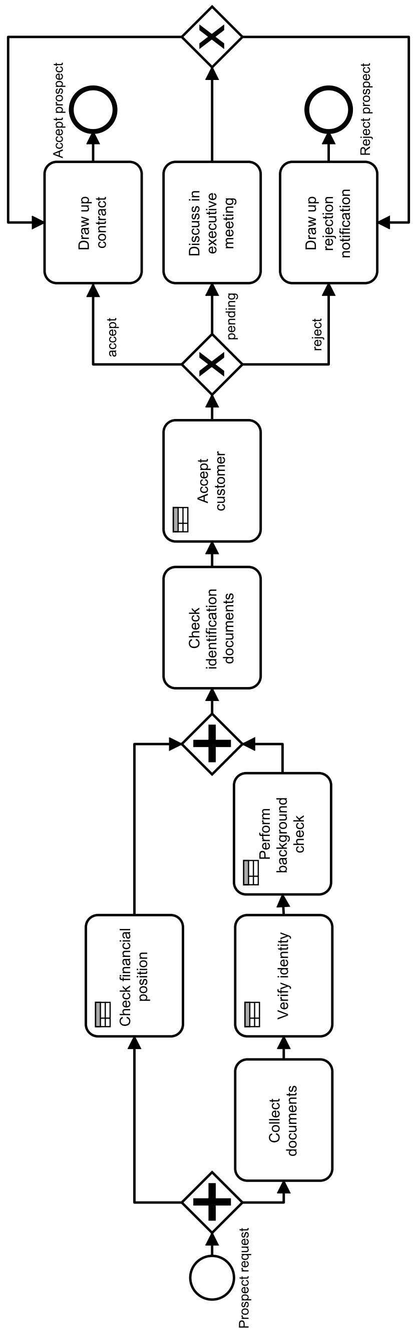


Figure 2.4: Iteration 2.

additional activity is introduced that requests valid documentation. In order to remedy **I6**, subdecisions that have an influence on the control flow of the process, must be modelled as decision activities in the process according to **P3.2**.

Figure 2.4 includes decision activities that do not produce relevant intermediate results for the process at hand and that do not influence the control flow of the process. More precisely, `Check Financial Position` and `Perform Background Check` decision activities are represented in the process model. If the decision activity does not refer to the top-level decision or does not provide an intermediate result that is used in the process, or does not influence the control flow of the process, it is not necessary to model that decision activity. Thus, Figure 2.4 contains **I4**. The obsolete decision activities can be excluded from the process model, as they can eventually be invoked by another higher level decision that is represented in the process model. Figure 2.5 provides a process model adhering to **P3.3**. Furthermore, representing all the decisions from the decision model in the process model opposes the declarative nature of the decision model, since by modelling all possible decision activities in the process, the decision execution is hard-coded in the process as well. This may lead to unnecessary delays in the process, e.g. the parallel gateway in Figure 2.4 forces the process to stop until both branches are joined before the process can continue further, while in reality this may not be ideal. In Figure 2.5, no such issues are present. Thus, Figure 2.5 conforms to **P3.1**, **P3.2** and **P3.3**, hence conforming to **P3** and only containing relevant subdecisions that influence the process in terms of data, intermediate results, and control flow.

2.7.4 Inclusion of decision requirement hierarchy

A consistent process model should respect the decision requirement hierarchy provided by the decision model. Figure 2.5 violates this condition, as the `Determine Risk Level` subdecision activity is located after the top-level decision activity `Accept Customer`. The decision `Customer Acceptance` *requires* the outcome of the subdecision `Risk Level`, and this hierarchy should be respected by the order of the decision activities in the process model. Decision activity `Accept Customer` will require an outcome of decision

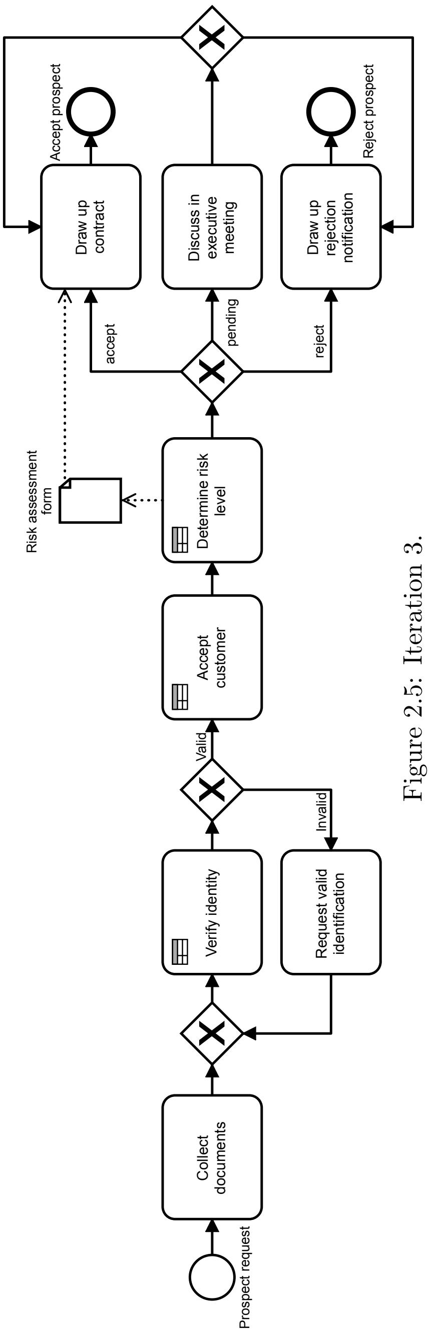


Figure 2.5: Iteration 3.

activity `Determine Risk Level` before the former can be executed successfully. This corresponds to **I5**. To restore the decision requirement order in the process model one incorporates **P4** and simply switches the two decision activities. Figure 2.6 provides a model that solves this inconsistency.

2.7.5 Inclusion of relevant data and advanced data management

In Section 2.4.2 we defined decision activities as activities that have an input and an output, with a logical connection between the two. Figure 2.6 does not respect these definitions as it violates **I7**. In Figure 2.6 three decision activities are present, yet none of them has the required inputs. Only decision activity `Determine Risk Level` shows an output in the form of a *Risk Assessment File*, while other decision activities don't exhibit any output data object. Hence, Figure 2.6 shows poor data management and decisions cannot be enacted without the proper data input. Decision activity `Verify Identity` is linked to the subdecision `Customer Identity Verification` in the decision model in Figure 2.1. The decision model reveals that the *Customer ID* is needed as input for this subdecision and the process model in Figure 2.6 does not provide this indispensable data management.

Figure 2.7 remedies the data management issues for all decision activities in the process model and links them to the relevant constructs in the decision model, hence conforming to **P5**. The activity classification is key to solving this problem. The decision model in Figure 2.1 requires input data, namely *Customer ID*, *Financial Statements*, *Financial Information* and *Public Records*. As explained in Section 2.4.2, input data is produced by administrative activities. Those activities have no input, but do produce output. In the process in Figure 2.7, the input data needed for a sound enactment are produced by administrative activities `Collect Documents`, `Request Valid Identification` and `Look Up Information`. The relevant data objects are then linked to the (sub)decision activities that exploit them as input data, e.g. decision activity `Verify Identity` uses *Customer ID*, produced by `Collect Documents` or `Request Valid Identification`. Each decision activity also produces an output

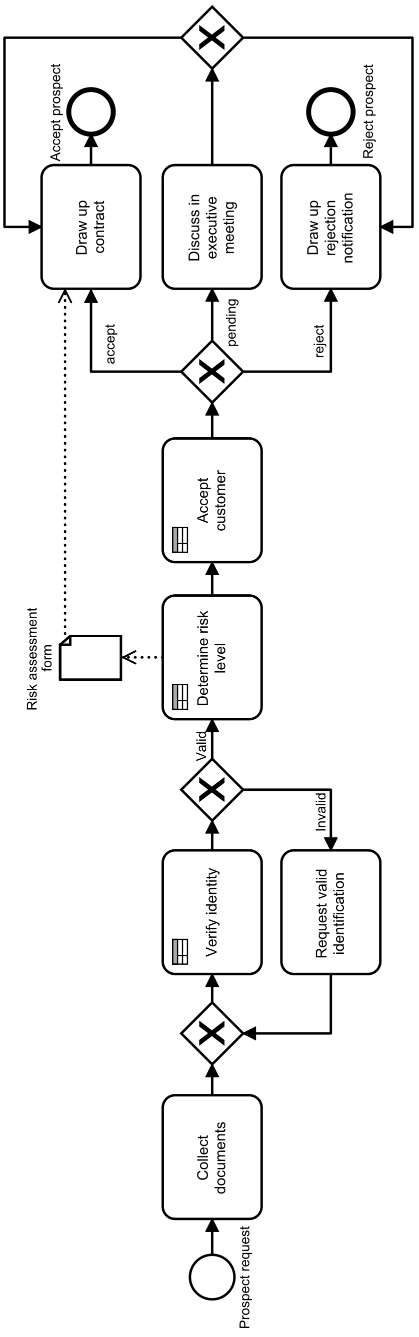


Figure 2.6: Iteration 4.

data object, as previously specified in the definition for decision activities.

The intermediate results of subdecision activities such as `Verify Identity` and `Determine Risk Level` are used in higher level decision activities as input, in accordance to the decision model. The intermediate result of decision activity `Verify Identity`, *Identity Verification*, is used in decision activity `Determine Risk Level`, together with the other input data required to enact the subdecision `Risk Level` in the decision model. Likewise, the intermediate result of decision activity `Determine Risk Level`, *Risk Assessment File*, is adopted as input for the decision activity `Accept Customer`, which corresponds with the top level decision `Customer Acceptance` in the decision model. The outcome of the top level decision is then used to determine whether or not to accept the customer and to draw up a contract if the customer gets accepted. A final activity classification from Section 2.4.2 refers to operational activities, i.e. activities that might have an input, but that produce no decision output. In Figure 2.7 `Draw Up Rejection Notification`, `Draw Up Contract` and `Discuss In Executive Meeting` are representatives of the operational activity classification.

2.8 Resolving Inconsistencies

This section deals with resolving inconsistencies and adhering to the principles of integrated modelling in a systematic way. In [5] soundness is defined to achieve **P1** for isolated decision points. **P1** and **P2** are rather straightforward: make sure that every decision outcome can be handled by the process flow and avoid hard-coding decision logic in processes. Principle **P3.2** is the complement of **P1**, as **P1** suggests that when a decision activity that impacts the control flow is modelled, all its outcomes should be taken into account by the control flow. On the other hand, **P3.2** determines that if a decision impacts the control flow of the process, it should be explicitly modelled as a decision activity in the process.

However, **P3.1**, **P3.3**, **P4** and **P5** require additional attention. Using the formal basis from the previous sections, we defined process-decision model consistency by two conditions in Definition

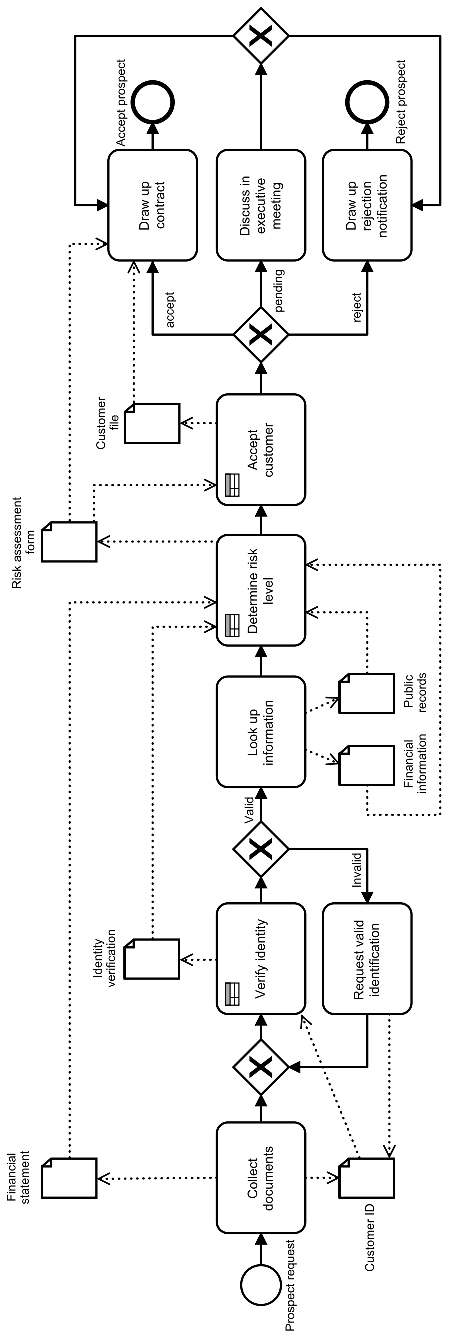


Figure 2.7: Iteration 5.

2.10:

1. No intermediate results of non-invoked subdecisions are used.

2. Each (sub)decision invoked in the process, must be guaranteed to be invocable at that stage of the process.

The first condition in Definition 2.10 refers mainly to **P3.1** and **P3.3**, while the second condition acknowledges **P4** and **P5**. In the following subsections we will address each of these necessary conditions.

2.8.1 Resolving the use of intermediate results

Violations against the first condition for consistency of Definition 2.10 can be resolved in the following steps:

1. Identify the subdecisions producing intermediate results that are used in the process, both in terms of data objects and control flow.

2. Add these subdecisions to the process model in the form of decision activities in the correct hierarchical order.

3. Do not include the remaining subdecisions into the process.

Note that this solution incorporates **P3.1**, **P3.3** and even **P4**. In the running example of the Belgian Accounting firm case in Section 2.7, we identified that the `Risk Level` decision produces a *Risk Assessment form* as intermediate result and that said result is needed later on in the process. After identifying a subdecision with its relevant intermediate result, we incorporated the subdecision in the process under the `Determine Risk Level` decision activity and consequently we also took the decision hierarchy into account by placing the `Determine Risk Level` decision activity before the decision activity `Accept Customer` that represents the top level decision and that requires the outcome of the subdecision `Risk Level`.

The topological order derived from Figure 2.1 induces that `Customer Identity Verification` $\leq$ `Risk Level` and that `Risk Level` $\leq$ `Customer Acceptance` according to Property 1. Thus, these decisions can be represented in the process model by their respective decision activities as long as they respect the topological order provided in the DRD. The first model adhering to all three steps with regard to resolving intermediate results in Section 2.7 is the one in Figure 2.6. Here, the intermediate result of the relevant subdecision is identified and the corresponding decision activity is incorporated in the process, while respecting the topological order of the DRD and while excluding process-irrelevant subdecision activities.

2.8.2 Resolving invocability inconsistencies

The second condition provided in Definition 2.10 revolves around invocability of (sub)decisions. In order to invoke a certain decision in the process through a decision activity, all relevant data needed for invoking that decision and its subdecisions must be available. Additionally, if subdecisions of the decision that is invoked are modelled within the process by means of decision activities, the intermediate results of the subdecisions must be readily available as well. Note that this condition mainly refers to **P4** and **P5**, i.e. the decision requirement hierarchy will ensure the availability of intermediate results (**P4**) and **P5** additionally assures the presence of other indispensable input data. Hence, violations against the second condition for consistency of Definition 2.10 can be resolved in three simple steps:

1. Apply the topological hierarchy of decisions from the decision model to the order of modelled decision activities in the process.

2. Ensure that all input data present in the decision model is present in the process model as well, either as external data or as internal process data.

3. For every decision activity in the process make sure that all input data and intermediate results of its subdecision activities in the process are linked to the decision activity as input data objects.

Consider Figure 2.7 and decision activity `Determine Risk Level` which represents the decision `Risk Level` from the decision model in Figure 2.1. According to the decision model, the `Risk Level` decision requires intermediate results from its subdecisions. In order to enact `Financial Position Check` the *Financial*

Statements and *Financial Information* input data is needed. In Figure 2.7, this data is linked to the decision activity as it was generated in the process earlier on by two administrative activities, `Collect Documents` and `Look Up Information`. To enact the second subdecision, `Background Check`, the *Public Records* input data is necessary as well. This too is provided by an administrative activity of the process and linked to the decision activity `Risk Level`. Finally, the intermediate result of the `Customer Identity Verification` subdecision of `Background Check` is required as well. This intermediate result was produced earlier by the process, since the `Verify Identity` decision activity was invoked with the necessary *Customer ID* input provided by an administrative activity. This intermediate result, i.e. *Identity Verification*, is linked as an input data object to decision activity `Determine Risk Level`. Hence, `Determine Risk Level` has all input data and/or intermediate results necessary to invoke the `Risk Level` decision and the process can proceed. Thus, ensuring that all necessary input data and intermediate results for a decision are available before the decision is invoked, resolves the invocability inconsistency.

2.9 Conclusion and Future Work

This chapter provides insights and principles for integrated process and decision modelling. While most previous works approach the problem in a straightforward way, i.e. only considering decision points and containing decisions to one specific place in the process, we analyse decisions holistically as they can span over multiple activities and even over the entire process. A DMN formalisation and classification of process activities is provided to connect decisions to processes. Next, based on the formalisation, inconsistencies are revealed. To remedy these inconsistencies, **Five** **P**rinciples for Integrated **P**rocess and **D**ecision **M**odelling (**5PDM**) were derived. The usefulness of **5PDM** is illustrated through a case from a Belgian Accounting firm. Additionally, a systematic stepwise approach towards consistent integration of processes and decisions was contributed. This approach relies on a sound management of intermediate results of decisions and on correctly matching the information requirements of decisions to process data.

In future endeavours we will investigate how the decision model can further aid in refactoring the process model. Additionally, decision making across distributed processes [14] in cooperative information systems is of particular interest for Internet of Things (IoT) application areas [78].

CHAPTER 3

An Illustration of 5PDM

"New opinions are always suspected, and usually opposed, without any other reason but because they are not already common."

An Essay Concerning Human Understanding
— *John Locke*

This chapter presents sections 3 and 4 of the following paper:

Faruk Hasić, Johannes De Smedt, Jan Vanthienen. An illustration of Five Principles for Integrated Process and Decision Modelling (5PDM). `FEB Research Report KBI_1717 (KU Leuven)`, Leuven (Belgium), 1-8, 2017.

Abstract. This chapter discusses an application example of integrated process and decision modelling guidelines aimed at consistently integrating process and decision model. The process models are depicted by the Business Process Model and Notation (BPMN), while the decision model is represented using the newly introduced Decision Model and Notation (DMN) standard of the Object Management Group (OMG). The example in this chapter revolves around an integration of a blank loan approval process with its underlying bank loan decision model. The process model is iteratively adopted to conform to the proposed **Five P**rinciples for integrated **P**rocess and **D**ecision **M**odelling (**5PDM**), thus rendering the process consistent with the underlying decision model.

3.1 An Example from Literature

In this section we provide a process model that is inconsistent with its underlying decision model and consequently we apply the proposed integrated modelling guidelines, rendering the process model consistent with the decision model. Figure 3.1 depicts a loan approval decision hierarchy consisting of a top level decision *Loan Approval* and five subdecisions. A corresponding bank loan approval process model is provided in Figure 3.2. The process is based on the model provided in *Fundamentals of Business Process Management* by [44] on page 91. For the sake of representation we have simplified the original process to fit on one page in this chapter.

As will become clear, the process in Figure 3.2 is inconsistent with the decision model in Figure 3.1. The inconsistencies are highlighted in Figure 3.2 and the **5PDM** principles needed to remedy the inconsistencies are indicated in the figure as well. We will concisely explain the application of the integrated modelling principles to the inconsistent process in Figure 3.2, thus rendering a process depicted in Figure 3.3 that is consistent with its underlying decision model represented in Figure 3.1.

First, notice that all six decision from the decisions model in Figure 3.1 are represented by their corresponding decision activities in the process in Figure 3.2. Employing Principle **P3** teaches us which decisions to explicitly model as decision activities in the process. More precisely, **P3.2** tells us to include decisions that lead to a change in control flow as decision activities in the process. Clearly, this is the case for the `Assess eligibility`, `Approve loan`, and `Negotiate payment` decision activities. They all divert the control flow of the project depending on their outcome and hence they are relevant to the process and should remain in the process. Also the `Check application form completeness` decision activity should remain in the process. According to Principle **P1**, all necessary decision outcomes, relevant to the process, should be modelled in the control flow following the decision activity representing the decision in the process. Assuming that `Check application form completeness` can have two possible outcomes: a positive outcome if the application form is complete and a negative one if that is not

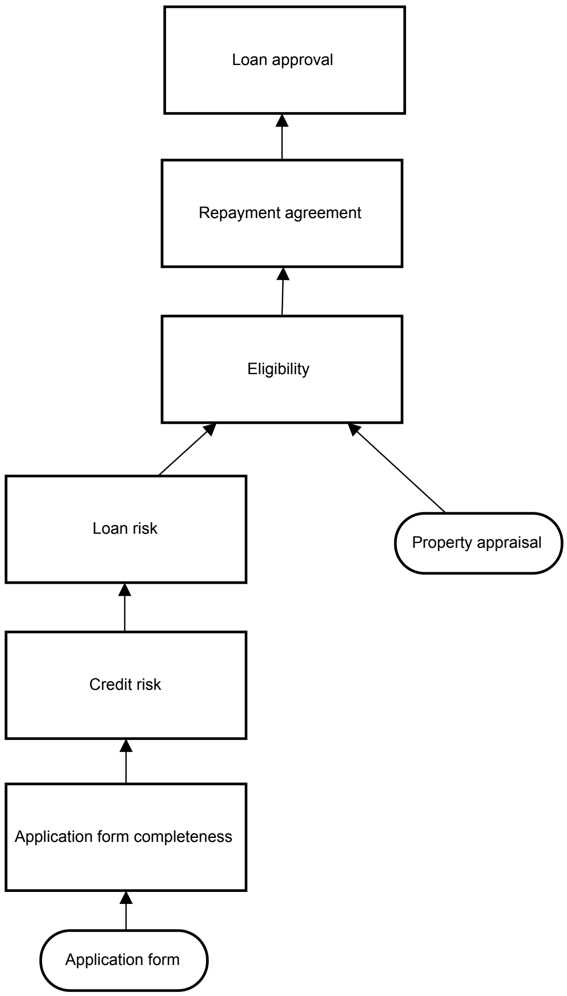


Figure 3.1: Decision model for a bank loan approval.

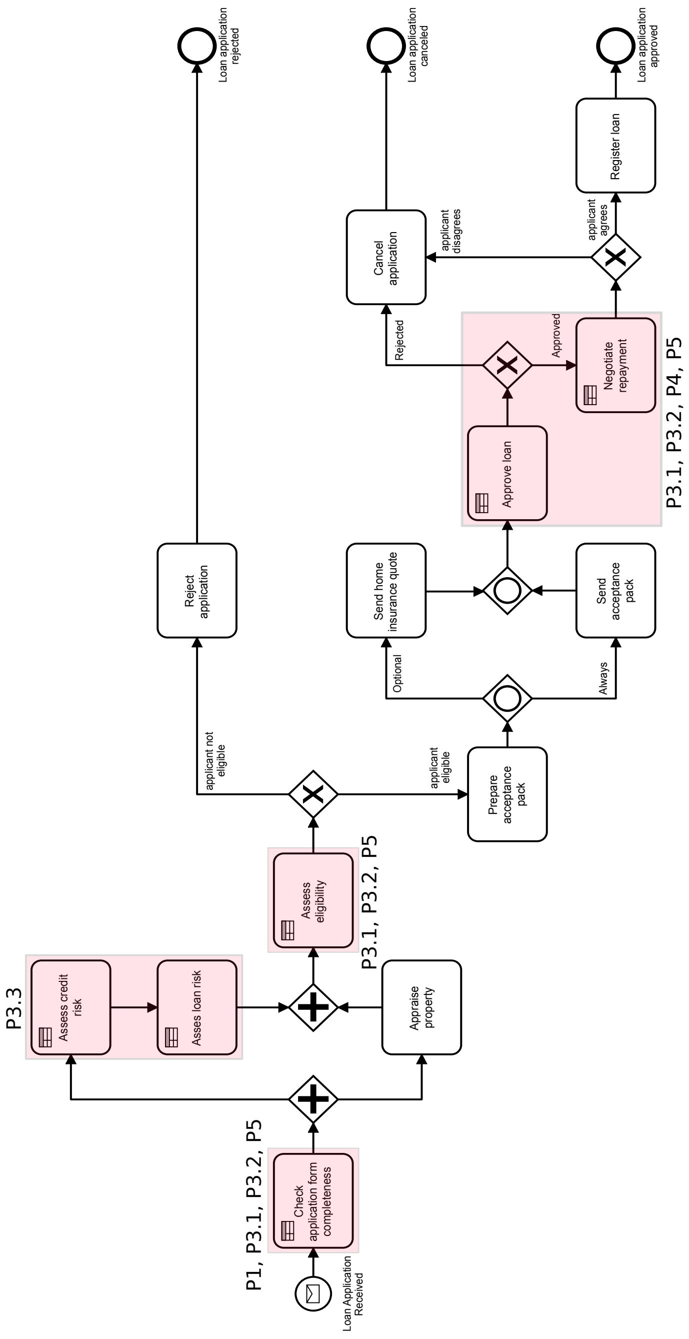


Figure 3.2: Process model for a bank loan approval.

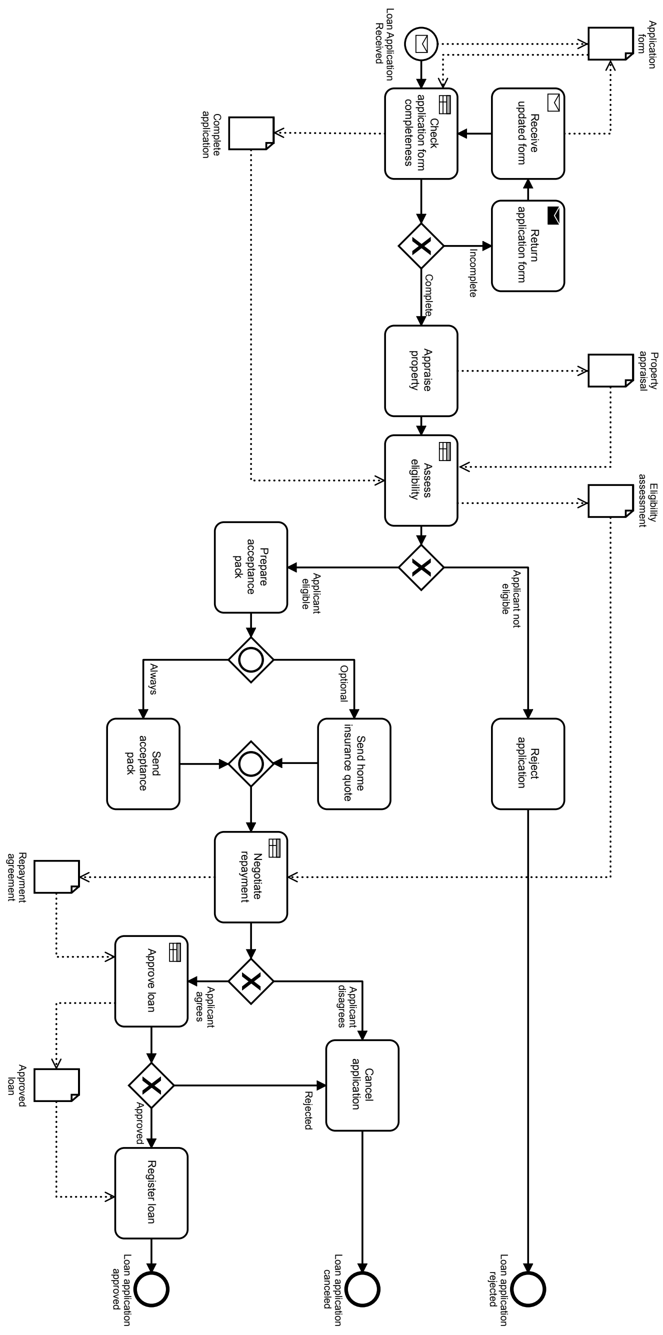


Figure 3.3: Process model for loan approval consistent with the decision model.

the case, i.e. the application form is incomplete. The latter should divert the process back to the decision activity, and the process can proceed to a subsequent stage once that decision activity reaches a desirable outcome. Hence, `Check application form completeness` will divert the process flow back to the decision through a loop, and hence the decision activity affects the process and should remain in the process according to **P3.2**. In the consistent model in Figure 3.3 these decision activities are therefore still present.

On the other hand, the decision activities `Assess credit risk` and `Assess loan risk` do not impact the process directly in the stage where they are modelled. Since higher level decisions of these subdecisions are present in the remainder of the process model, these particular decision activities need not explicitly be modelled within the process, as stated by Principle **P3.3**. Hence, these decision activities are not present in the consistent process model in Figure 3.3.

Note also that the process in Figure 3.2 does not conform to the topology of the decision model in Figure 3.1: the `Approve loan` and `Negotiate payment` decision activities are not ordered according to the decision requirements hierarchy present in the decision model. While in the decision model in Figure 3.1 the *Repayment agreement* is a subdecision of the *Loan approval* top level decision, the decision activity `Approve loan`, pertaining to decision *Loan approval*, precedes the decision activity `Negotiate payment`, pertaining to decision *Repayment agreement*. That way, according to the process in 3.2, the *Loan approval* decision is forced to enact before the prerequisite enactment of the *Repayment agreement* subdecision. This violates Principle **P4**, which states that the decision requirements hierarchy present in the decision model should be respected when modelling the corresponding decision activities within the process. Hence, decision activities `Approve loan` and `Negotiate payment` should switch places, as remedied according to **P4** in the process in Figure 3.3.

Now we have identified which decision activities should be discarded from the process model, and which should be present in the process model and in what hierarchical order. Given that the decision activities left in the process model of Figure 3.3 are representing decisions pertaining to the same decision model in

Figure 3.1, there exists a data and decision outcome dependency between those decision activities, as stated by Principle **P5**. The higher level decision activities will need the decision outcome of the lower level decision activities in order to enact properly. Thus, the data propagation of decision outcomes between related decision activities was taken into account according to Principles **P3.1** and **P5**. By definition, all decision activities have input data and output data. Taking into account the decision hierarchy in the decision model, a sound propagation of data can be achieved by connecting the decision outcomes of lower level decision activities to decision inputs of higher level decision activities. This data propagation management is modelled in the consistent process model in Figure 3.3 making sure that every decision activity has the correct input data as stated by the decision model.

Likewise, every decision activity has output data that can be used by a higher level decision activity, or simply any other operational activity within the process. For instance, in Figure 3.3 the outcome of the `Assess eligibility` decision activity, the *Eligibility assessment*, is propagated as input to the higher level decision activity `Negotiate payment`, in accordance to the decision requirements present in the decision model in Figure 3.1. This indeed conforms to Principle **P5**. Similarly, the outcome of decision activity `Approve loan`, the *Approved loan* data object, is used as input for the operational activity `Register loan`. This is indeed in accordance with Principle **P3.1**, which states that decision activities whose outcomes are used in process should explicitly be modelled in the process. In this case, decision activity `Approve loan` represents the top level decision of the decision model in Figure 3.1. Thus, this decision activity influences the process in multiple ways: from the control flow perspective by diverting the process and forcing it to reach a certain conclusion, and from the data perspective providing subsequent activities with the necessary input to enact properly.

3.2 Conclusion

To conclude, by applying the **Five P**rinciples for integrated **P**rocess and **D**ecision **M**odelling (**5PDM**), we have rendered the inconsistent process in Figure 3.2 to be consistent with its underlying decision model in Figure 3.1. The consistent process model is depicted in Figure 3.3. As such, this chapter provides an illustration of the 5PDM guidelines on an example based on the model provided in *Fundamentals of Business Process Management* by [44] on page 91. The example illustrates the usefulness of the 5PDM framework and it shows that consistent integration should rely on a profound data management of intermediate results of subdecisions and on correctly matching process data necessary for decision enactment to the information requirements in the decision model.

CHAPTER 4

Comparing BPMN to BPMN + DMN for IoT Process Modelling

"The duty of the man who investigates the writings of scientists, if learning the truth is his goal, is to make himself an enemy of all that he reads, and, applying his mind to the core and margins of its content, attack it from every side. He should also suspect himself as he performs his critical examination of it, so that he may avoid falling into either prejudice or leniency."

Doubts Concerning Ptolemy
— *Hasan Ibn al-Haytham (Alhazen)*

This chapter was published as follows:

Faruk Hasić, Monique Snoeck, Estefanía Serral Asensio. Comparing BPMN to BPMN + DMN for IoT Process Modelling: A Case-Based Inquiry. `35`$^{\text{th}}$ `ACM/SIGAPP Symposium on Applied Computing (SAC)`, Brno (Czech Republic), 2020.

Abstract. The network of interconnected devices that compose the Internet of Things (IoT) continues to expand. Business processes are starting to take advantage of IoT by adapting to the physical environment or by automating process tasks. The Business Process Model and Notation (BPMN) specification has been employed in numerous studies to include IoT devices and resources. While aggregating low-level IoT data into process-relevant data is of paramount importance for IoT processes, BPMN may not be the best approach to model this data aggregation. Decision Model and Notation (DMN), however, is a recently introduced standard which is inherently used to aggregate low-level information into high-level information. This makes DMN a promising match for modelling context data aggregation in IoT processes. Therefore, this chapter examines the modelling of IoT processes by comparing the standard BPMN approach and the combination of BPMN and DMN. Three cases with increasing need for context aggregation are modelled according to both techniques, leading to an analysis of the capability of the approaches to support IoT processes in terms of high-level context-awareness, scalability and complexity, flexibility, and decision logic reusability. We demonstrate that in cases where a need for complex context aggregation decision logic is present, the combination of BPMN and DMN provides the required support, even for the complex cases, and performs better than BPMN on its own.

4.1 Introduction

IoT can be used to facilitate both businesses and individuals in their daily endeavours. The application areas are vast and diverse, from smart home environments [121] to medical monitoring systems [72]. Business processes can benefit from IoT by understanding their execution context and adapting to it. This integration of IoT and business processes has enjoyed significant attention in literature, in particular several extensions to the BPMN [105] standard have been proposed to accommodate this integration [21]. However, these extensions often miss some of the support that the BPMN standard provides, such as the executability of the modelled processes, the interchangeability of models between tools, or process model integration with other languages. Hence, sticking to standards may provide a number of advantages compared to a language extension.

IoT processes often expose a need for context-awareness and context aggregation [90]. A recently introduced standard for decision modelling, i.e., DMN [106], is inherently used to aggregate information, as constructing a DMN model corresponds to aggregating low-level information into higher-level information, thus, creating a hierarchy of decision information with different levels of granularity. This makes DMN a promising match for modelling context data aggregation.

Given the inherent need for context aggregation in IoT processes, this chapter examines the modelling of IoT processes by comparing the standard BPMN approach and the combination of BPMN and DMN. For this purpose, we model and compare three exemplar IoT process cases taken from literature. To avoid the need for developing extra support for language extensions, throughout the chapter, we stick to both BPMN and DMN as standardised modelling languages, so as to maintain the advantages that come with those standards regarding executability, interchangeability, and integration of process models. From this endeavour, we evaluate the ability of the BPMN + DMN versus the BPMN-only approach for dealing with context aggregation, the scalability and complexity of processes, the flexibility of decision and processes, and context aggregation decision logic reusability.

This chapter is structured as follows. Section 4.2 provides background information and related work. Section 4.3 revolves around three IoT process case studies with increasing needs for context aggregation. Section 4.4 evaluates the advantages of BPMN + DMN according to the ability to aggregate context information, scalability, flexibility, logic reusability, and the adherence to principles for cognitively effective modelling notations. Finally, Section 4.5 concludes and discusses future research.

4.2 Preliminaries and Related Work

This section provides preliminary information and related work on IoT, IoT processes, their modelling, and DMN.

4.2.1 IoT

IoT refers to the ever-growing network of interconnected devices. Generally speaking, an IoT environment is populated by two kinds of devices: sensors and actuators. Sensors serve as a mechanism to capture reality and are therefore the source of IoT data towards the system. Actuators on the other hand operate in the reverse direction, i.e. they change the reality by activating a physical reaction that is initiated by the system. A typical sensor is for example a luminosity sensor. On the other hand, a typical actuator can be an electrical motor. Data collected by IoT devices is considered to be part of the context information, i.e. the information describing the conditions under which the system operates. Relevant context information might for instance include the location of an object.

4.2.2 IoT processes

IoT processes focus on incorporating IoT into process management [150]. This IoT and business process integration can be twofold. By definition, IoT business processes are enacted in a dynamic and highly connected physical environment. Since IoT is a technology that can be used to digitalise the context of a system, the process is granted the capability of both understanding its context and changing it through IoT devices. Understanding context is a difficult task which requires the inference of high-level context from the the low-level data caught by IoT devices: they need to be aggregated in order to render context information that is meaningful for the process [35]. Different levels of context abstraction can be defined [90]. As shown in Figure 4.1, the raw data collected from the IoT devices is at the lowest level of abstraction. This level of abstraction can be handled by a context monitor [121] which captures the data coming from the sensors and makes sense of it. A carbon level sensor can for instance provide a data stream of *420, 425, 465, 8357*. The context monitor observes these data measurements and stores them in a system as follows: the carbon level in a room is *325* at a specific time. This is simple context information which can be stored for further use within the business processes. This way, the business process can access the latest relevant data which can be interpreted for further process execution.

However, often a higher level of abstraction is needed to make meaningful decisions within the process. The simple context information needs to be aggregated even further to render more abstract and complex context information. For instance, a carbon level of *8357* can be considered as dangerous if people are present in the room. Hence, smart room sensor information, obtained from a presence sensor and a carbon level sensor, can be aggregated into ventilation settings to safeguard air quality, thus inferring a higher-level meaningful context. We refer to this level of abstraction when discussing context aggregation in this chapter, i.e., the aggregation of simple context information into more complex contexts. The transformation from raw sensor data to simple context information is out of scope for this chapter, and we refer to works such as [121] for this matter. On another side, IoT devices can be used to automate tasks, e.g., opening the curtains in a smart home. The integration of IoT in BPMN models has been, among others, discussed in [150].

4.2.3 IoT process modelling languages

BPMN is the most extended language to model IoT business processes, see e.g., [21, 130, 150]. Most of these extensions focus on introducing new symbols to discriminate tasks and events that are IoT-related from regular BPMN elements. However, they often do not consider the aggregation of simple context to more complex context information. A lack of standardisation can be immediately observed as well: extensions diverse from one another and are normally not compatible with the standard BPMN, preventing them from taking advantage of the additional support it provides, like the executability of the modelled processes, or its integration with other languages. The combination of different standards as such could contribute towards solving these issues.

4.2.4 DMN

DMN [106] is a recently introduced decision modelling standard that consists of two levels. First, the decision requirement level which depicts the dependencies between elements involved in the decision model. Second, the decision logic level, which presents ways to specify the underlying decision logic, usually modelled through decision tables. DMN employs rectangles to depict decisions and ovals to represent data input. DMN has been adopted in recent literature [42, 59]

Note that DMN modelling is inherently used to aggregate information, as constructing a DMN model corresponds to aggregating low-level information into higher-level information, thus creating a hierarchy of decision information with different levels of granularity. This makes DMN a great match to model context data aggregation [56, 90]. Take for instance the example explained in Section 4.2.2, which can easily be modelled as shown in Figures 4.8 and 4.9, where a DMN model aggregates the lower-level

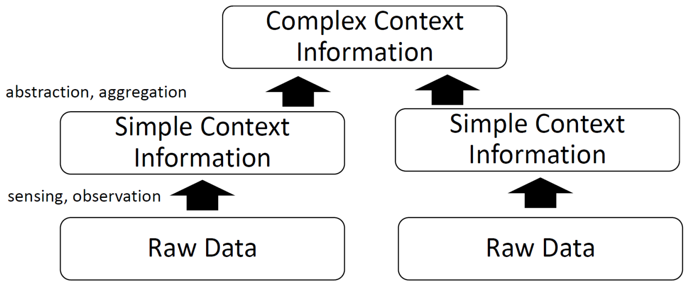

Complex Context Information
abstraction, aggregation
Simple Context Information
Simple Context Information
sensing, observation
Raw Data
Raw Data

Figure 4.1: Hierarchy and aggregation of context [90].

contexts into higher-level information.

The combination of BPMN and DMN has already been studied by several authors [13, 37, 62, 66, 67, 123, 150]. However, none of these works discuss the use of this combination to describe IoT processes without further extensions. This combination achieves separation of concerns by having separate process and decision models. Instead of modelling decision constructs in the process model, the decisions are externalised and encapsulated in a separate DMN decision model. The key to this process and decision integration lies in identifying and modelling process elements that function as intermediaries of the decision model within the process model. Some of these process elements, i.e., decision activities, invoke a decision from the DMN decision model by providing the correct data input requirements set for that decision, and as such, invoking the decision logic stored in the decision model. Subsequently, the decision model returns a decision outcome to the process for further interpretation.

4.3 IoT Processes Modelling Cases

This section provides three exemplar IoT process cases from literature. We adhered to purposive sampling for selecting the cases [47]. Purposive sampling revolves around carefully selecting cases expecting that each case provides relevant information for the problem at hand. We queried Google Scholar with the search string *('IoT' OR 'Internet of Things') AND ('scenario' OR 'case') AND ('process' OR 'BPMN' OR 'Domain Model' OR 'Petri net')*. We obtained 153 IoT cases. We provide the complete list of cases online[2]. Since we are interested in IoT processes where aggregation of context information is a challenge, we selected three cases with increasing order of context aggregation necessity, ranging from simple context aggregation to more complex context aggregation. For each case, models are built according to standard BPMN on the one hand, and standard BPMN and DMN on the other. Notice that for BPMN models, we only use standard elements that are executable. Non-executable modelling constructs, e.g., complex gateways, are omitted, as we aim to exploit the executability

[2]`https://github.com/SAC2020/SAC/blob/master/ListOfCases.xlsx`

support of the standard as well.

4.3.1 Case 1: smart transportation

Case description. Consider the case described in [129], about a refrigerated truck transporting perishable food, i.e. food that is not fully frozen but that is likely to spoil if not adequately cooled. The internal air exchanges heat with the outside environment as heat is conducted into the truck from the ambient air and from solar radiation on the outside of the truck. Given that outside temperature conditions influence the temperature inside the truck as well, the cooling system should proactively provide adaptable temperature management to maintain adequate inside air temperature based on outside weather conditions. For that purpose a temperature sensor can be placed on the outside of the truck, and the cooling system can automatically adapt to the outside temperature by either increasing or decreasing the refrigeration.

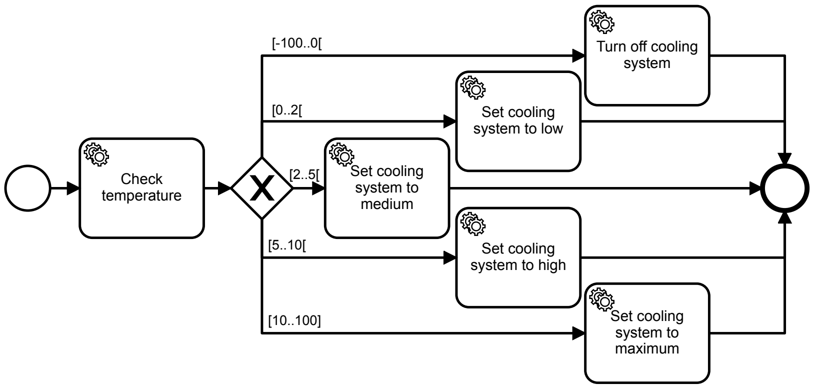

Check temperature
[-100..0[
[0..2[
[2..5[
[5..10[
[10..100]
Turn off cooling system
Set cooling system to low
Set cooling system to medium
Set cooling system to high
Set cooling system to maximum

Figure 4.2: An IoT-enhanced smart transportation process.

BPMN IoT process. The IoT-enhanced business process is depicted in Figure 4.2. All the tasks in the process are IoT tasks represented as automated service tasks. Once the temperature has been checked, the exclusive gateway evaluates the temperature and decides on the cooling level that must be set to guarantee that the perishable food will reach the destination unspoiled. The logic

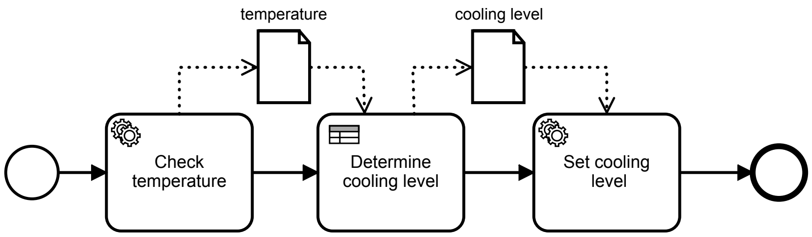


Figure 4.3: Decision-aware smart transportation process.

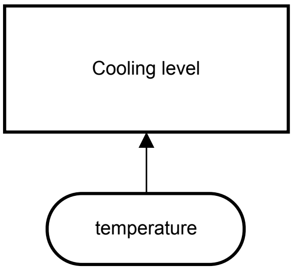
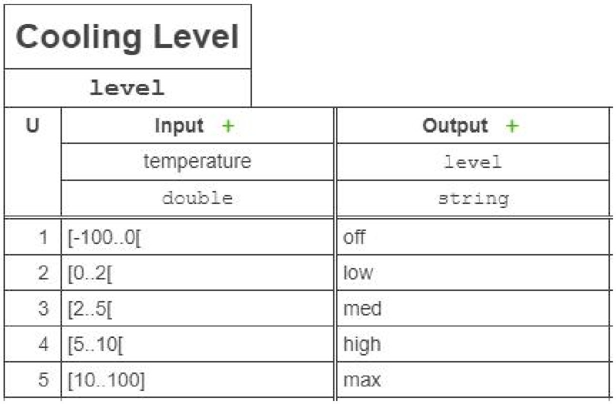


Figure 4.4: Smart transportation DRD.

needed to decide on the cooling level is embedded in the control flow.

BPMN+DMN IoT process. According to integrated BPMN and DMN modelling, decisions are not mapped to control flow elements. Rather, decisions are externalised into a separate DMN decision model that can be invoked by the process. Consider Figure 4.3 representing the decision-aware process. The Check temperature service task retrieves the temperature. With this data a decision task invokes the smart transportation decision model from Figure 4.4. The decision logic of the `Cooling level` decision is modelled in Figure 4.5. As such, the process provides the decision model with the relevant input data, i.e. the *temperature*, after which the decision model evaluates the decision rules and returns a *cooling level* decision outcome to the business process. This cooling level is subsequently used in the `Set cooling level` service task to adjust the cooling level within the refrigerated truck.



Cooling Level

level

U	Input +	Output +
	temperature	level
	double	string
1	[-100..0[	off
2	[0..2[	low
3	[2..5[	med
4	[5..10[	high
5	[10..100]	max

Figure 4.5: Cooling level decision table.

4.3.2 Case 2: smart ventilation

Case description. Consider the convention centre case presented in [130], which consists of a large conference room that can be booked for various events. The convention centre is concerned about the air quality in the conference room when events take place. For that purpose a smart ventilation system is installed. Presence sensing is used to assess whether the room is being occupied and air quality sensors monitor the CO_2 concentration rate of the room. The ventilation system is slowed down when no presence is registered in the conference room, or when presence is detected and CO_2 concentration rates are acceptable. This is done to ensure low energy consumption. If on the other hand presence in the room is detected together with high concentrations of CO_2, the ventilation in the conference room is increased to safeguard air quality and conference participants' comfort.

BPMN IoT process. The BPMN for this case was modelled in [130]. The presence of participants in the conference room and the CO_2 concentrations are checked, and based on the evaluation of the data received, the ventilation in the room is either increased or decreased. After the scheduled ending time of the conference, the process again checks for the presence of the participants to decide whether the ventilation system can be switched off. We have adapted the original process to use service task instead of

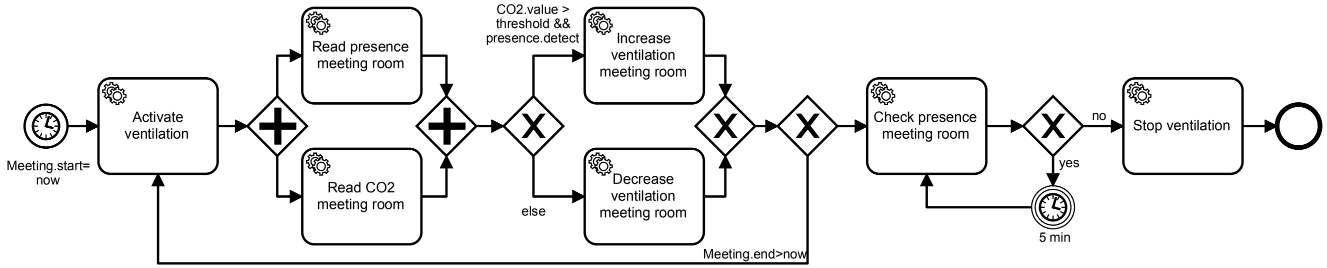

Meeting.start=
now
Activate ventilation
Read presence meeting room
Read CO2 meeting room
CO2.value > threshold && presence.detect
else
Increase ventilation meeting room
Decrease ventilation meeting room
Meeting.end>now
Check presence meeting room
no
yes
5 min
Stop ventilation

Figure 4.6: Smart ventilation convention centre core IoT process, modified from [130].

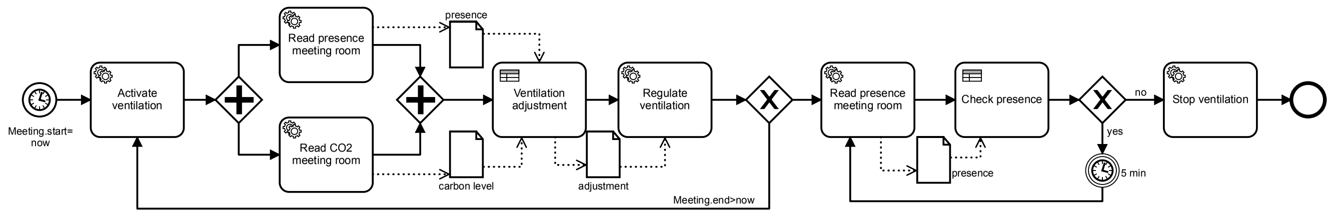


Figure 4.7: Smart ventilation convention centre decision-aware process.

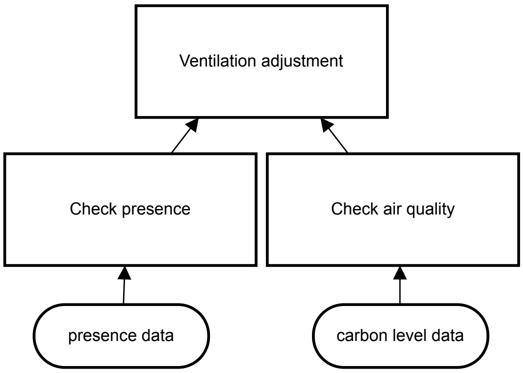


Figure 4.8: Smart ventilation decision model.

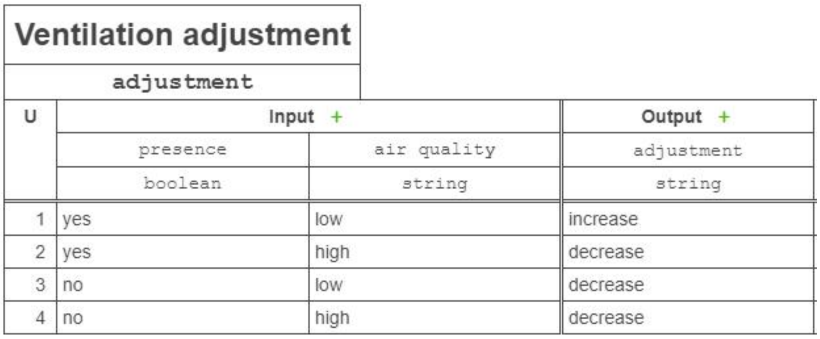
wireless sensor tasks in order to be as uniform as possible across the three cases (see Figure 4.6). In addition, we have corrected a few mistakes that the original model presented[3]. In particular, the original model suffered from a deadlock caused by the backward loop to the parallel gateway, as well as erroneous, overlapping and missing conditions. Figure 4.6 is therefore an adapted version of the process, so as to ensure a fair comparison.

BPMN+DMN IoT process. Figure 4.7 shows the equivalent decision-aware process. Notice that the decisions are externalised to the model in Figure 4.8, while the logic is modelled in the decision table of Figure 4.9.

4.3.3 Case 3: smart healthcare monitoring

Case description. Consider a patient health monitoring system for a person diagnosed with the Chronic Obstructive Pulmonary Disease (COPD). This case was presented in [72]. COPD is a disease that obstructs the lungs and the airflow and breathing of the patient. Acute attacks of the disease can happen. In that case the patient can experience uncomfortable complications such as fast breathing, a fast heart rate, hyperactive use of muscles, and a cold skin. It has been recognised that an IoT-based patient

[3]`https://github.com/SAC2020/SAC/blob/master/SmartVent.pdf`
See also Figure A.1 in Appendix A.



Ventilation adjustment

adjustment

U	Input +		Output +
	presence	air quality	adjustment
	boolean	string	string
1	yes	low	increase
2	yes	high	decrease
3	no	low	decrease
4	no	high	decrease

Figure 4.9: Ventilation adjustment decision table.

monitoring process can help increasing the life quality of the patient and decrease the risks that are inherent to the disease and multiple sensors and wearable technologies exist that can collect patient data relevant for the patient monitoring process [72]:

- Electrocardiogram (ECG) sensors monitor the heart.
- Respiratory sensors check the breathing rate.
- Skin temperature sensors monitor the skin temperature.
- Muscular Electromyography (EMG) sensors monitor the muscle activity.

All these sensors collect measurements on the patient’s health. Note that this case displays a high need for context aggregation, as a single sensor or even a few sensors combined are not enough to capture the COPD. For instance, the patient might take a walk outside in the winter and a sensor registers a low skin temperature. In that case, the patient is not necessarily suffering from COPD at that moment. However, an expert can build patient-specific decision rules to capture COPD in such a monitoring system. For instance, if the sensors register a low skin temperature, a short and fast breathing rate, together with a high heart rhythm, the monitoring process might decide that the patient is suffering an attack and running out of oxygen. In such a situation the process can trigger the administration of an oxygen mask to the patient. Less severe attacks can be remedied by using an inhaler.

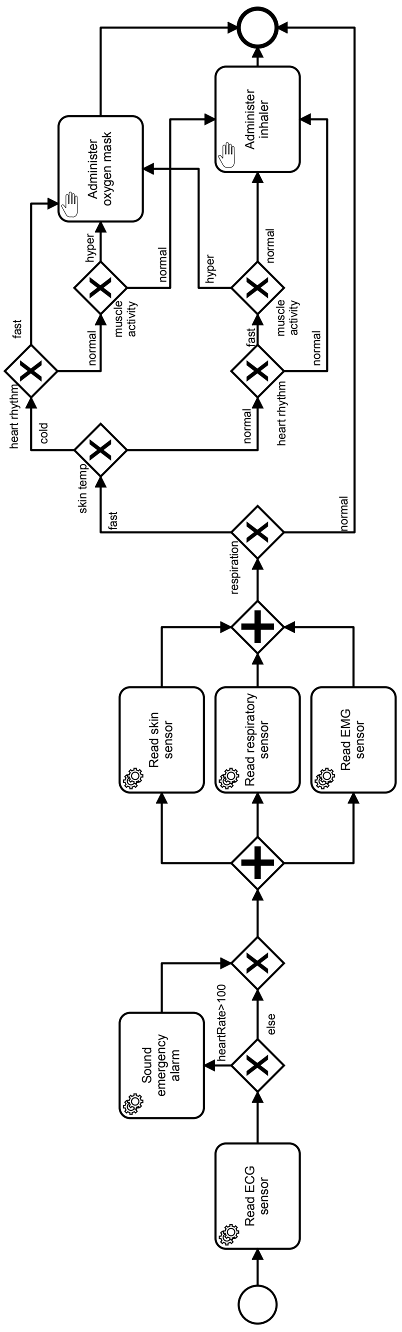


Figure 4.10: An IoT-enhanced COPD monitoring process.

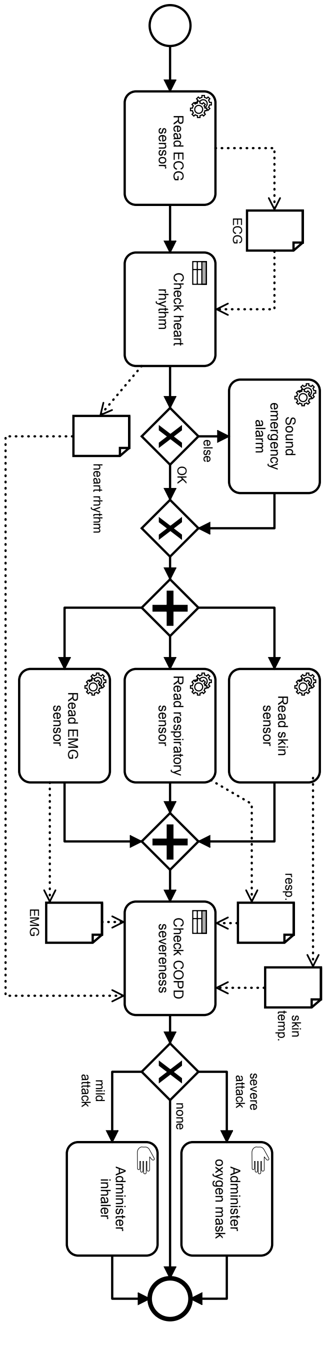


Figure 4.11: A decision-aware process model for a COPD monitoring IoT process.

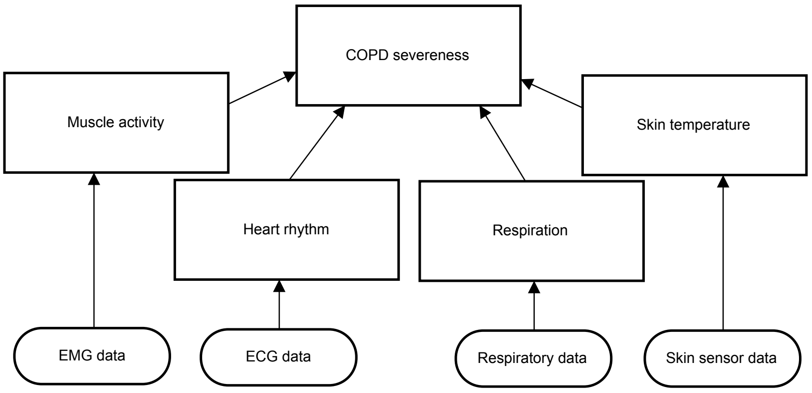
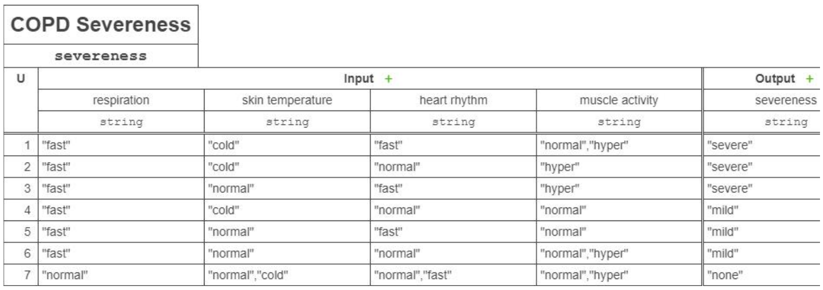


Figure 4.12: COPD severeness DMN model based on IoT data.


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 4.13: The COPD severeness decision table.

BPMN IoT process. The BPMN process is given in Figure 4.10. The process starts by reading the heart rhythm sensor and by sounding an emergency alarm if the heart rhythm was assessed to be worrisome, i.e., if the heart rhythm exceeds 100 beats per minute. This logic is embedded in the exclusive gateway following the `Read ECG sensor` service task. After the possible emergency alarm, three other sensors are consulted by the process, i.e. a respiratory sensor, a skin temperature sensor, and a muscle activity sensor. Following that, the values received from these sensors are scrutinised in the intricate exclusive gateway flows. Hence, the decision rules are hard coded into the control flow. Based on these decisions the process either terminates or selects one of the two treatments.

BPMN+DMN IoT process. A decision-aware COPD monitoring process is given in Figure 4.11. The underlying decision model based on IoT data is provided in Figure 4.12, while the decision logic of the top level `COPD severeness` decision is modelled in the decision table in Figure 4.13. Note that the process model in Figure 4.11 first enacts the `Check heart rhythm` business rule task which in turn invokes the subdecision `Heart rhythm` in the decision model in Figure 4.12. Afterwards, after a potential emergency alarm for heart problems, the process checks COPD severeness after which the process finishes by either administering an oxygen mask to the patient in severe cases, or administering an inhaler to the patient in mild cases, or without providing any treatment.

4.4 Does BPMN + DMN provide advantages over BPMN?

In this section, we compare the two approaches to model IoT processes in terms of their ability to aggregate context information, scalability and complexity, flexibility, and context decision logic reusability, as well as the adherence to principles for cognitively effective notations.

4.4.1 DMN and context aggregation

The ability of the proposed approach to render complex or high-level context from simple or low-level context information is a step of paramount importance in IoT systems [90]. Take for instance *Case 3* describing the smart healthcare monitoring scenario. Low-level data are collected by IoT sensors that monitor the patient, from which a simple context can be derived pertaining to the collected data. The low-level context derived from a single sensor does not provide adequate information for making decisions and carrying out the process. Instead, the low-level context information needs to be aggregated and processed to obtain high-level context information that is meaningful for decision making. When modelling with the standard BPMN approach, such as in Figure 4.10, this aggregation of low-level context into high-level context is ad hoc and unsystematic, i.e., the low-level context information is hard-coded into the control flow of the process, without any transformation or reasoning mechanism to infer a higher-level semantic context. In such a situation, one needs to analyse the control flow paths to arrive at an aggregation of low-level information that makes up the high-level context or meaning needed in the process, i.e., by following a sequence of gateways, the high-level context is revealed, such as the administration of an oxygen mask pointing towards a severe COPD attack. An approach to avoid embedding decision logic in control flow elements would be to use BPMN script tasks where the decision logic is embedded as script code within the script task. However, this is coding rather than modelling, and it raises similar challenges regarding complexity (the decisions are hidden in code and are hence difficult to understand) and flexibility of the decisions (changing decision logic would correspond to digging up script code and re-coding it). We provide script task examples for the cases online[4].

Since constructing a DMN model corresponds to aggregating low-level contexts into higher-level context information, a hierarchy of contexts at different levels of granularity is constituted. This was shown in for instance the aggregation of wearable sensor information into the severeness level of a disease in *Case 3* or the aggregation of smart room sensor information into ventilation settings in *Case 2*. Next to this aggregation of context, it is worthwhile to emphasise that DMN models make subdecisions explicit, such as the `heart rhythm`, `skin temperature`, `muscle activity`, and `respiration` subdecisions in *Case 3*. Hence the decision process is provided with better modularity, as higher-level, aggregated decisions provide information hiding about how lower-level subdecisions are taken. Furthermore, issues regarding overlapping and missing decision rules can more easily be checked and avoided in DMN than in ad hoc rule insertions in the control flow. This was exemplified in the presence of errors in the original model found in literature of *Case 2*, indicating that the correct capture of decision logic in a BPMN IoT process is not self-evident. Works that automatically detect missing and overlapping rules in DMN decision tables have been presented in literature [31].

[4]`https://github.com/SAC2020/SAC/blob/master/Script.zip`
See also Figures A.2-A.6 in Appendix A.

4.4.2 Scalability and complexity

Here we consider size-based metrics that have been proposed in literature to assess the scalability and complexity of the models developed in the previous section. The results are shown in Table 4.1. The DMN metrics TNR (total number of rules), NOD (number of decisions), and TNDO (total number of data objects) are taken from [58, 88]. These are the only size-based DMN metrics that have been proposed in literature. The DMN metrics are calculated for the entire DMN model, not just the top-level tables presented in this chapter. Lower-level decision tables are provided online[5]. BPMN metrics NOA (number of activities), NOG (number of gateways), and NOF (number of flows) are used by suggestion of [99], where guidelines for element count-based process metrics are proposed. In the case of a single process model where the logic has been hard-coded in the control flow, the number of gateways and control flow paths grows with the introduction of additional decision rules. In processes that rely on a vast amount of decision rules, this leads to a lot of branching conditions and control flow elements, leading to spaghetti-like processes. Note that the decision process could also be modelled in a separate BPMN process model and invoked by the process requiring the decisions by the introduction of BPMN call activities. An example of *Case 3* where the decision logic is externalised into a separate callable decision process is provided online[6]. However, in this decision process, the ordering of the condition tests leads to a much longer decision path than when modelling contracted decision tables such as in Figure 4.13. This is exemplified in the number of exclusive gateways and control flow paths in the callable decision process.

Integrated BPMN and DMN modelling tends to decrease process model complexity according to model-size based metrics. This is shown in Table 4.1, where it is clear that, for all three cases, the number of gateways and flows decrease. However, integrated BPMN and DMN modelling introduces new complexity through the adoption of additional decision models. This complexity is

[5]`https://github.com/SAC2020/SAC/blob/master/Tab.zip`
See also Figures A.7-A.12 in Appendix A.

[6]`https://github.com/SAC2020/SAC/blob/master/Call.zip`
See also Figures A.13 and A.14 in Appendix A.

Table 4.1: Size-based complexity metrics.

Case	Model	TNR	NOD	TNDO	NOA	NOG	NOF
Case 1	BPMN				6	1	12
	BPMN+DMN	5	1	1	3	0	4
Case 2	BPMN				7	6	19
	BPMN+DMN	8	3	2	8	4	17
Case 3	BPMN				7	10	27
	BPMN+DMN	15	5	4	9	5	20

represented by the TNR, NOD, and TNDO metrics in Table 4.1, thus indicating that part of the complexity is offloaded from the process model to the decision model when using BPMN+DMN. Nevertheless, using BPMN+DMN facilitates the separation of the process and decision modelling concerns. Moreover, with a DMN decision table one can check for overlapping and missing rules, as well as reorder and contract the rules in the decision table [31], leading to a shorter decision path in the process than in the case where decision rules are hard-coded into a decision tree-like structure in the process control flow.

4.4.3 Flexibility

Flexibility, also called maintainability, is determined by two factors: *understandability* and *modifiability* [48]. In the Constructive Cost Model (COCOMO) [23], *understandability* is part of the maintenance adjustment factor (MAF), while *modifiability* is expressed in terms of the software or model parts that need to be changed (i.e., elements added and elements removed). Integrated BPMN and DMN modelling facilitates *understandability* of decisions by modelling them in a separated and structured form, i.e., interconnected decision tables, and thus provides possibilities to detect missing and overlapping rules [31]. This was exposed in *Case 2*, where the original process model contained overlapping and missing rules due to unsystematic decision modelling that is embedded in the control flow of the process. *Modifiability*, on the other hand, refers to the cost of making changes to existing models and can be measured in terms of *edit distance*, i.e., the minimal number of add and delete operations needed to transform a legacy schema into a new schema. Note that with increasing numbers of input variables and variable values, the decision logic becomes more complex, as the number of input combinations increases. While in *Case 1* the BPMN model is still readable, we have seen that in *Case 2* and *Case 3* the logic becomes increasingly difficult to understand, verify, and modify.

Changing decision logic is different when the logic is externalised in a separate decision model than when it is embedded in the process flow. In the former case the decision model needs to be updated. In the latter case, the process flow and branching conditions in the process model need to be adapted. This can be a burden with complicated control flow sequences. Take *Case 3* and suppose that rule 6 in the decision table of Figure 4.13 needs to be split up into two rules, one with preconditions *("fast","normal","normal","normal")* with outcome *"mild"*, and one with preconditions *("fast","normal","normal","hyper")* with outcome *"severe"*. This corresponds to an edit distance of 3: deleting an existing rule and adding two additional rules in the decision table of Figure 4.13. The corresponding process model in Figure 4.11 does not need to undergo any changes as it can still access the logic in the decision model and according to the decision outcome, continue towards process conclusion. If on the other hand such a change needs to take place in the BPMN model of Figure 4.10, where the logic is hard-coded in the control flow of the process, the process would need to undergo redesign, rendering an edit distance of 5: an existing flow would need to be deleted, and an additional exclusive split and three corresponding flows (one incoming and two outgoing) need to be introduced to allow for the decision rule to be split up. As explained in Subsection 4.4.2, this is due to the fact that DMN allows for rule verification possibilities, as well as the reordering and contraction of decision rules in the decision tables [31]. This way, rules can be modelled in a more compact form, leading to shorter decision paths than in the case where rule conditions are hard-coded in the process flow. Note however that, depending on the decision logic, compacting and simplifying decision tables is not always possible. If such a possibility exists, modifying the rules in the compact form requires less changes than modifying the elements in the long hard-coded decision paths of the BPMN process flow.

4.4.4 Reusability of the decision logic

The same decision logic may be used in different processes, especially when inferring high-level knowledge, i.e., complex context information as shown in Figure 4.1. Therefore, reusability of decisions is of paramount importance for IoT processes, especially when rendering complex context from simple context information.

In the case when using a single BPMN process model to incorporate both the process and decision concerns, the derivation of complex context information is hard-coded and confined to a local place in one process model, as was shown in the cases presented in this chapter. However, as explained in Subsection 4.4.2, the decision process could be modelled in a separate BPMN process model and invoked by the process requiring the decisions by the introduction of BPMN call activities. Partial decision logic can be modelled as a call activity as well, used in an overarching call activity. That way, modularity can be achieved. However, next to the creation of longer decision paths as explained in Subsection 4.4.2, additional issues surface as a result of this approach. First, decisions tend to be declarative, and modelling them in a procedural BPMN model goes against this property. Second, even if the decisions were modelled as a decision process that can be requested by a BPMN call activity, contraction and verification possibilities are lacking.

On the other hand, integrated BPMN and DMN models consider decisions, and in the case of this chapter aggregated context decision logic, in a separate decision model, thus making aggregated context decision logic reuse for other processes and systems a possibility as those other processes and systems can call upon a decision from the decision model. This way, the derivation of complex context information is reusable across processes, systems and applications, rather than hard-coded and confined to a decision point in one process model. Thus, DMN offers a modular decision requirements hierarchy which allows for decision logic reuse and verification possibilities: modelling, reusing, and checking the decision rules is easier and more efficient by means of DMN than by hard-coding a decision tree into process control flow paths.

Table 4.2: Adherence to principles for designing cognitively effective notations. Precise definitions of principles: see [98].

Principle	**Adherence**
Semiotic clarity	A gateway refers to a single decision point, whereas DMN offers constructs for complex decisions.
Perceptual discriminability	In BPMN+DMN decisions are easier to discriminate in the model due to the rule task symbol.
Semantic transparency	The rule task symbol in BPMN+DMN implies a meaning and is semantically transparent.
Complexity management	DMN provides better modularity and decision hierarchy than hard-coded BPMN decision flows.
Cognitive integration	BPMN+DMN supports integration of information from different models, i.e., BPMN and DMN models.

4.4.5 Adherence to the physics of notations

The work in [98] defines a set of principles for designing cognitively effective notations. The principles form the *Physics of Notations* theory as they focus on the perceptual properties of notations. The principles are derived from theory and empirical research and can be used to evaluate and compare existing visual notations. We employ the relevant principles to compare BPMN to BPMN+DMN. A comparative overview is given in Table 4.2. According to these principles, we can see that BPMN+DMN is a better choice.

4.5 Conclusion and Future Work

This chapter examined the modelling of IoT processes by comparing the standard BPMN modelling approach and the combination of BPMN + DMN. For this purpose we have modelled three different cases with increasing needs for context aggregation with both approaches. A trade-off exists between modelling IoT processes exclusively in BPMN versus modelling them in an integrated BPMN and DMN fashion. The former avoids introducing additional types of models, and thus additional complexity and longer learning curves. However, modelling IoT processes in such a manner impairs the scalability and flexibility of the process, as well as decision logic reuse. Furthermore, an important advantage of BPMN + DMN is the ability to perform modular, explicit, and reusable context aggregations through DMN decision modelling. The advantages of BPMN + DMN become increasingly noticeable in cases of complex context aggregation decision logic, as was shown in the three cases presented in this chapter. While in the simple context aggregation in *Case 1*, the model is still readable using BPMN, we have seen that in *Case 2* and *Case 3* the logic becomes increasingly difficult to understand, verify, and modify. In those cases, the combination of BPMN and DMN is particularly beneficial given the advantages induced by the separation of the process and decision modelling concerns, i.e., process scalability, process and decision flexibility, decision logic reusability, and last but not least, the ability to modularly aggregate context information.

In future work we will empirically assess the complexity, understandability, and maintainability of standard BPMN models and integrated BPMN and DMN models in IoT settings in order to provide an empirical assessment as an additional form of evaluation for the work presented in this chapter. Furthermore, we plan to compare the BPMN extensions proposed in literature with the combination of BPMN and DMN, focusing on the pragmatics of the conceptual modelling quality framework.

Part III

Decision as a Service (DaaS)

CHAPTER 5

Decision as a Service (DaaS): A Service-Oriented Architecture Approach for Decisions in Processes

"I too play with symbols [...]; but I play in such a way that I do not forget that I am playing. For nothing is proved by symbols [...]; things already known are merely fitted; unless by sure reasons it can be demonstrated that they are not merely symbolic but are descriptions of the ways in which the two things are connected and of the causes of this connection."

Letter to Joachim Tancke, Gesammelte Werke, XVI
— *Johannes Kepler*

This chapter was published as follows:
Faruk Hasić, Johannes De Smedt, Seppe vanden Broucke, Estefanía Serral Asensio. Decision as a Service (DaaS): A Service-Oriented Architecture Approach for Decisions in Processes. IEEE Transactions On Services Computing, *article in press*, 2020.

A preliminary version of this paper was published in:
Faruk Hasić, Johannes De Smedt, Jan Vanthienen. A Service-Oriented Architecture Design of Decision-Aware Information Systems: Decision as a Service. 25th International Conference on Cooperative Information Systems (CoopIS), Rhodes (Greece), 353-361, 2017.

Abstract. Separating decision modelling from the processes modelling concern recently gained significant support in literature, as incorporating both concerns into a single model impairs the scalability, maintainability, flexibility and understandability of both processes and decisions. Most notably the introduction of the Decision Model and Notation (DMN) standard by the Object Management Group provides a suitable solution for externalising decisions from processes and automating decision enactments for processes. This chapter introduces a systematic way of tackling the separation of the decision modelling concern from process modelling by providing a Decision as a Service (DaaS) layered Service-Oriented Architecture (SOA) which approaches decisions as automated and externalised services that processes need to invoke on demand to obtain the decision outcome. The DaaS mechanism is elucidated by a formalisation of DMN constructs and the relevant layer elements. Furthermore, DaaS is evaluated against the fundamental characteristics of the SOA paradigm, proving its contribution in terms of abstraction, reusability, loose coupling, and other pertinent SOA principles. Additionally, the benefits of the DaaS design on process-decision modelling and mining are discussed. Finally, the DaaS design is illustrated on a real-life event log of a bank loan application and approval process, and the SOA maturity of DaaS is assessed.

5.1 Introduction

Recent business process management literature moves towards accommodating decision management into the paradigms of *Separation of Concerns (SoC)* [16, 52, 61, 63, 65, 132] and *Service-Oriented Architecture (SOA)*. This implies externalising decisions and encapsulating them into separate decision models, hence implementing decisions as externalised services. Literature proposes several conceptual decision service platforms and frameworks [22, 96, 151], as well as ontologies [89]. Industry has adopted this trend, as several decision service management systems have appeared, e.g., SAP Decision Service Management [118]. This separation of concerns provides a plethora of advantages regarding understandability, maintainability, and flexibility of both the business process and the decision models [16, 52, 60, 61, 63, 79, 132].

A recently introduced decision modelling standard, the Decision Model and Notation (DMN) [106], has enjoyed significant interest in literature [61, 78, 104, 108]. 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. DMN aims at providing a clear and simple representation of decisions in a declarative form and offers no decision resolution mechanism of its own. Rather, the invoking context, e.g., 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. This makes DMN particularly interesting for a SOA, as DMN is independent of the applications and the invoking context.

However, there is no clearly and formally defined SOA design dealing with DMN decision services in business processes. This leads to violations against the SoC and SOA paradigms in previously produced models in research, where decisions tend to be embedded or hard-coded within the process, and where decision logic tends to be duplicated in both the process and decision models. This chapter aims at bringing DMN to the service-orientation paradigm in order to exploit the benefits of SOA characteristics in terms of maintainability, scalability, understandability, and flexibility. Thus, the contribution of this chapter is a layered design framework and a formalisation of its key concepts to abstract the decision logic and decouple it from the process layer according to the SoC [52] and SOA paradigms. The layered architecture consists of a process layer, a service layer, and a decision layer. The two latter are connected to the former through a decision service interface. The separated layers are presented in Figure 5.2. The evaluation of the contribution is fourfold:

1. The proposed DaaS design is assessed in terms of fundamental SOA principles, proving its merit in terms of service discoverability, loose coupling, standardisation, location transparency, abstraction, statelessness, longevity, reusability, and composability.

2. The implications of the DaaS design on process and decision modelling and mining are conferred in terms of scalability, maintainability, flexibility and understandability of both the processes and the decisions.

3. Additionally, we illustrate the DaaS design on an event log containing information on a real-life bank loan application and approval process. We show from the real-life event log that decisions are invoked as services, in compliance with the principles of SOA and that the DaaS design aids in conducting decision service compliance verification for processes. As such, we show that the mining of decision services can be achieved as illustrated by the example at the end of the chapter.

4. Finally, we assess the maturity of the DaaS SOA design using a state-of-the-art SOA maturity model.

This chapter is structured as follows. Section 5.2 constitutes a related work section while Section 5.3 outlines the methodology of the chapter. Section 5.4 provides formal definitions for key concepts needed for the understanding of the layered SOA. In Section 5.5 the layered architecture design is established and elucidated and in Section 5.6 the proposed Decision as a Service architecture is evaluated against the core characteristics of SOA design. In Section 5.7 the implications of the proposed design for integrated process and decision modelling and mining are discussed in terms of advantages and disadvantages. Section 5.8 illustrates the DaaS approach on a real-life enriched event log, thus showing that decision services can be mined, and providing opportunities for decision service compliance verification. Next, Section 5.9 assesses the maturity of the proposed design according to a state-of-the-art SOA maturity model. Section 5.10 discusses limitations of the approach and finally, Section 5.11 concludes.

5.2 Related Work

The *SoC* paradigm is already well-established in the software modelling and design domains [52, 81, 103]. With the introduction of DMN, the paradigm is introduced in the Business Process Management (BPM) domain as well, effectively shifting the domain towards a SOA, by representing business decisions as externalised services. Most modelling and mining approaches in literature still breach the SoC between process control flow on the one hand, and data and decision aspects on the other. Consequently, issues concerning maintainability, scalability, reusability, and understandability arise [61, 66], given the fact that most decisions are hard-coded within the process flow.

The intersection between data and processes, classical data mining and process mining is rapidly gaining traction in literature [62, 93, 122, 132, 133, 134]. Separating multi-perspective modelling and mining tasks proves to be beneficial in multiple ways, as long as the separation and interaction between the models is conducted in a sound and consistent way [16, 61, 66, 132]. Some literature on decision service platforms exists. In [22] decisions are approached from an organisational process control flow perspective rather than a data and decision management perspective. Other works recognise that decision logic and process logic should indeed be separated, as application logic is specified in terms of processes while decision rules specify conditions to adapt the application behaviour [96, 151]. However, these works consider simple decisions and business rules rather than holistic and intertwined decision models, i.e., modular subdecisions that are part of a holistic decision that spans across the whole process execution span and that is not confined to a single decision point in the process.

Furthermore, the works discuss general requirements and solutions, rather than a standardised approach towards decision management. Decisions did receive attention in the process mining domains, as researchers have utilised DMN to automatically discover decisions from event logs that are compatible with the discovered processes. However, these works define decisions in specific locations within a process known as decision points, i.e., exclusive gateways splitting the control flow of a process [12, 32]. Seminal work on mining decisions independently from process control flow is presented in [122], where decisions can span across the entire process execution span rather than being embedded in a single decision point of the process. This leads to the discovery of a holistic decision model that is decomposable and reusable across the process, thus, introducing a form of decision modularity and composition that is inherent to service-orientation.

Externalising decisions and setting them up as services is not the first example of adapting the SoC and SOA paradigms in the domain of processes: SOAs were already applied for business processes and for the interaction between business rules and processes [51, 89, 101, 118, 140, 141]. However, business rules are very granular and atomic and do not provide modularity, and aggregation and abstraction possibilities. DMN decisions, however, are an aggregation of rules and data and they provide abstraction from individual business rules by aggregating them into modular decision nodes. Thus, with DMN, a comparable approach can be applied to decisions by implementing decisions as externalised services, which we call **Decision as a Service (DaaS)**. Processes, or other concerns, can invoke those decision services on demand by providing the relevant input data to the service through an interface. The invocations do not necessarily need to be at classical decision points or gateways. Rather, they can occur anywhere in the process. We call this **Decision on Demand (DoD)**. Such an approach aims at capitalising on the benefits of SOA, such as loose coupling, service standardisation, abstraction, and composability.

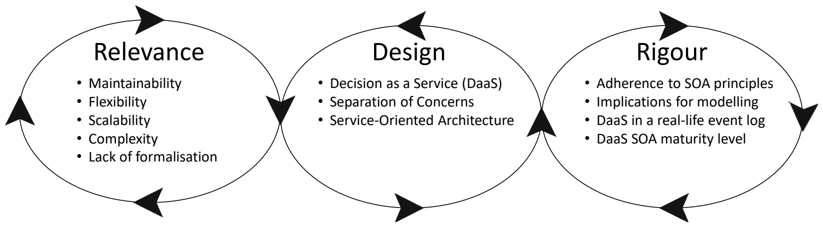

Relevance
• Maintainability
• Flexibility
• Scalability
• Complexity
• Lack of formalisation
Design
• Decision as a Service (DaaS)
• Separation of Concerns
• Service-Oriented Architecture
Rigour
• Adherence to SOA principles
• Implications for modelling
• DaaS in a real-life event log
• DaaS SOA maturity level

Figure 5.1: Overview of the followed cycles of the design science methodology.

5.3 Methodology

This chapter follows a design science approach [73], structured along three different cycles to obtain an artifact, being the SOA-based **Decision as a Service (DaaS)** design. Figure 5.1 provides an overview of the followed methodology.

First of all, during the *relevance* cycle we have identified the problem of inefficient use of decisions within processes and as a result the issues that arise regarding maintainability, scalability, flexibility, and complexity and understandability of both decisions and processes in Sections 5.1 and 5.2. We have argued that these are the relevant issues tackled when separating concerns in modelling endeavours through the use of the separation of concerns and SOA paradigms. Based on previous work of the authors, as outlined in the related work section above, and relevant literature - both academic and from industry - it was noted that there is no clearly and formally defined approach towards decision services in processes, and that from previously produced models in research multiple violations towards the separation of concerns and SOA paradigms were committed. Thus, we have built and evaluated the solution artifact in the form of a Decision as a Service (DaaS) design against core SOA and SoC principles during the *design* cycle (Sections 5.4 and 5.5). Next, the artifact, i.e. the proposed *Decision as a Service (DaaS)* design, which will be outlined in Section 5.5, was validated according to the *rigour* cycle against the key principles of Service-Orientation (Section 5.6). Additionally, the effects of the proposed solution were evaluated against the key problems identified in the relevance cycle (Section

5.7). Furthermore, this work aims at bringing the proposed design to the body of literature on decision and process modelling and mining by exhibiting the artifact on a real-life enriched event log from industry, i.e., an enriched event log pertaining to a bank loan application and allocation process. It is demonstrated in Section 5.8 that decision data propagation within the log of a process indeed exhibits the invocation of decision logic through variable shifts throughout the process. As a final evaluation, the design was assessed in terms of its maturity in a state-of-the-art SOA maturity model.

5.4 Preliminaries

In this section, we provide a formalisation for key DMN concepts needed for the understanding of the **Decision as a Service** architecture, which will be discussed in the following sections. We adhere to the definitions provided in [66] and extend them to represent decision services as well. The DMN standard employs rectangles to depict decisions and subdecisions and ovals to represent data input. The decision logic is usually represented in decision table form.

Definition 5.1. A decision requirement diagram DRD is a tuple (D_{dm}, ID, IR) consisting of a finite non-empty set of decision nodes D_{dm}, a finite non-empty set of input data nodes ID, and a finite non-empty set of directed edges IR representing the information requirements such that $IR \subseteq (D_{dm} \cup ID) \times D_{dm}$, and $(D_{dm} \cup ID, IR)$ is a directed acyclic graph (DAG).

The term *decision* can have a number of meanings. According to the DMN specification a decision is the logic used to determine an output from a given input. Meanwhile, in process modelling a decision is an activity or the act of using the decision logic, e.g. the business rule task in BPMN. Another common meaning is that a decision is the actual result, which we call the output of a decision, or simply the decision result. For the case of Decision as a Service, a decision is defined as follows:

Definition 5.2. A decision $d \in D_{dm}$ is a tuple (I_d, O_d, L), where $I \subseteq ID$ is a set of input symbols, O a set of output symbols and

L the decision logic defining the relation between symbols in I_d and symbols in O_d.

In case of decision tables, I and O contain the variables of the input and output elements respectively, and L is the table itself, i.e. the set of decision rules present in the table. Note that, since a DRD is a DAG, $I_d \cap O_d = \emptyset$. In DRDs these decisions d_i are represented by the decision nodes $D_i \in D_{dm}$. We will use D to refer to both a decision and its representing node in a DRD.

According to the DMN standard, a decision requirement diagram can be an incomplete or partial representation of the decision requirements in a decision model. The set of all DRDs in the decision model constitutes the exhaustive set of requirements. The information contained in this set can be combined into a single DRD representing the decision requirements level as a whole. The DMN standard refers to such a DRD as a decision requirement graph (DRG). We expand the notion of a DRG, in such a way that a DRG is a DRD which is self-contained, i.e. for every decision in the diagram all its requirements are also represented in the diagram.

Definition 5.3. A *DRD* is a decision requirement graph *DRG* if $\cup_{DRD \in M}(D_{dm}, ID, IR)_{DRD} = (D_{dm}, ID, IR)^{DRG}$, with M being the set of all DRDs. It holds that $IR^{DRG} \subseteq ({D_{dm}}^{DRG} \cup ID^{DRG}) \times {D_{dm}}^{DRG}$, and $({D_{dm}}^{DRG} \cup ID^{DRG},\ IR^{DRG})$ is a directed acyclic graph (DAG).

From Definition 5.3, it is clear that every decision D in the DMN model has a unique decision requirement graph DRG_D with D as its single top-level decision. A DRG contains exactly all information requirements of its top-level decisions. Hence, only one DRG exists with D as its single top-level decision. We use DRG_D to denote this DRG. Furthermore, all the decisions in the DRG, except the top-level decision, are consequently subdecisions of the top-level decision. In other words, the top-level decision requires these lower-level subdecisions.

Definition 5.4. A decision D' is a subdecision of decision D if and only if it is part of DRG_D, but not D itself.

This order of decisions and subdecisions can be defined by using the property that DRDs are directed acyclic graphs, from Definition

5.1. From this property we know that each DRD has a topological order. The concept of topological orders is closely related to partial orders.

Property 1. *The topological order of a DRD induces a partial order $\leq$ on the decisions contained in the DRD.*

5.5 Decision as a Service (DaaS)

Separating the decision modelling concern from the process modelling concern implies modelling in two separate models or layers [61, 63, 66]. In Figure 5.2, the **Decision as a Service** layered architecture is presented through an example of a customer acceptance process with its corresponding customer acceptance decision model. The bottom layer depicts the *processes layer*, while at the top the *decision layer* is represented. In the service-oriented approaches, the services are implemented offering a single decoupled point of entry to the services. That way, the bottom layer, i.e. the process layer, only needs the information regarding the point of entry, or more specifically the *interface*, in order to invoke the higher-level layers. This single point of entry provides an abstraction specifying how clients should interact with the decision services.

In Figure 5.2 the *service layer* is implemented as the connection between the *process layer* and the *decision layer*. The communication between the *process layer* and the *service layer* is bridged by the *interface*. Consequently, the processes are only aware of the *interface* and agnostic about the underlying *service layer* and *decision layer*. Thus, to invoke the services and the decisions the processes simply need to keep information regarding the *interface* and not regarding the higher level layers.

To formally define the interface and the decision services we first need to define the input requirement set of a decision as follows:

Definition 5.5. The decision input requirement set $dirs_D$ of a decision D is the set of all sets of input data which are sufficient to invoke D. $dirs_D$ contains sets of input data directly or indirectly required by D. The largest set in $dirs_D$ is the set of all input data nodes for which there exists a path to D in DRG_D. The smallest set in $dirs_D$ is D's input set I_D.

$dirs_D$ is constructed inductively by the following rules:

- $I_D \in dirs_D$
- For all $s \in dirs_D$ if there is an $i \in s$ such that $i \in O_{D'}$ for some D' in DRG_D, then $s \setminus \{i\} \cup I_{D'} \in dirs_D$.

As expressed in the last bullet point of the definition, the input requirement set is constructed by exploring the whole subtree of the DRD with root D. Each decision in a DRD has its own output set, as formalised in Definition 5.2. As seen in Figure 5.2, a decision service is used to invoke a decision from the decision model.

Definition 5.6. A decision service DS_D of a decision D is a tuple (s_D, O_D), where $s_D \in dirs_D$ is a set of input data sufficient to invoke the decision D and O_D the output set representing the decision outcomes of D.

The decision service is somewhat of a proxy for a part of the decision layer. As such, it can be argued that a subset of the DRG corresponds to a decision service and that decision services are in essence an abstraction of the decision model. The decision logic is encapsulated in the decision model and when a decision service is derived from the decision model, the logic that pertains to that service can be inferred from the decision model against which the service is defined. As such, the logic is kept in one layer, avoiding issues regarding decision logic duplication, inconsistency, and maintenance if the underlying logic in the decision model were to evolve, i.e., to undergo changes.

Note that multiple decision services can be defined for a single decision D, depending on which input data set s_D, as defined and constructed in Definition 5.5, is used to access the decision layer. Consider the decision model in the decision layer of Figure 5.2. For Decision `Background Check` two decision services can be defined: DS_{BC1} with tuple $(\{O_{CIV}, pr\}, O_{BC})$ where O_{CIV} is the output of Subdecision `Customer Identity Verification`, which serves as the input for Decision `Background Check` and *pr* is the *Public Records* input file; and DS_{BC2} with tuple $(\{cid, pr\}, O_{BC})$ where *cid* is the *Customer ID* input data object required for the decision enactment of Subdecision `Customer Identity Verification` and consequently the `Background Check` decision, and *pr*, i.e. the *Public Records* input data object. Which decision service will be activated depends on the input data set provided through the interface by the process in the process layer. Hence, the interface will steer the information towards the suitable service that is able to invoke the required decision based on the input data set received from the process. Thus, a decision service's interface is the combination of its input requirement set and its output set. Decision interfaces can be defined as in Definition 5.7.

Definition 5.7. The interface IF_D of a decision service DS_D is defined as a tuple $(dirs_D, O_D)$, where $dirs_D$ is the input requirement set and O_D the output set of the underlying decision D.

Now we have formally defined the decision layer, the service layer and the interface layer present in Figure 5.2. What is left is to define how the process layer and the different interactions with the service interface. Decisions in processes do not surface solely as the driver of control flow. Rather, they both encompass the routing, i.e. because of decision outcomes that steer toward a certain activity tailored towards supporting its output, and the changes in the data layer of the process as well. The latter introduces numerous types of activities that are representatives of the *decision* model in the *process* model:

Definition 5.8. The input and output data variables of business activities are defined as follows:

- $I : A \rightarrow V$, function assigning activities which receive input of a certain variable,
- $O : A \rightarrow V$, function assigning activities which deliver output for a certain variable.

This enables the construction of the following activity types:

1. **Operational activities ((no) inputs, no outputs):** do not have any influence on the process' decision dimension and only act as a performer of a specific action that is tied to that specific place in the control flow. They might serve as the conclusion of a decision. They are provided with the decision inputs needed, which are not used further in the process, $A_o = \{a \in A \mid O(a) = \emptyset, \}$.

2. **Administrative activities (no inputs, outputs):** have the purpose to introduce decision inputs into the process, $A_a = \{a \in A \mid I(a) = \emptyset \wedge O(a) \neq \emptyset\}$.

3. **Decision activities (inputs, outputs):** serve a true autonomous decision purpose as they transform decision inputs into a decision outcome, $A_d = \{a \in A \mid I(a) \neq \emptyset \wedge O(a) \neq \emptyset\}$.

Note that it holds that $A_a \cup A_o \cup A_d = A$.

With the activity classification in mind, we can now make the connection with decisions in business processes and decision models. A decision in a business process can be defined as follows:

Definition 5.9. A decision in a process model, $d^a \in D_{dm}$ is a tuple $(I_{d^a}, O_{d^a}, L_{d^a})$, where $a \subseteq A_d$, $I_{d^a} \subseteq I(a)$, $O_{d^a} \subseteq O(a)$ and $L_{d^a} \subseteq L$.

This last definition connects a decision activity with a decision and it shows than one decision activity can be tied with multiple decisions. The latter implies that, within an event log, the same activity can make different decisions, i.e., changes in variable values, and can be represented as different decision nodes within a decision model, as well as different activity types. This interpretation of how activities are present in process models is the main difference with other decision mining and modelling techniques, who keep the one-to-one mapping of activities and decisions.

We have formally defined elements from all three layers in the DaaS design in Figure 5.2: the decision layer, the decision service layer, the decision service interface, and the constructs from the process layer that are relevant for decision services and decision enactment in processes. Whether a process will correctly call upon a decision service depends on the information that the process itself provides to the decision service interface. In order to invoke a decision service successfully and unambiguously the input data objects needed for the invocation of the underlying decision need to be both complete and correct. This is defined in the decision **Service Adherence Criterion (SAC)** in Definition 5.10, which states when a process is compliant to the decision service it wishes to invoke.

Definition 5.10. The Service Adherence Criterion (SAC): A process fully adheres to the decision service DS_D of decision D if and only if at the time of invocation, the process has internally produced and/or externally received all $x \in s_D$ with $s_D \in dirs_D$ and provided all $x \in s_D$ to the decision service interface IF_D. Only then will the decision service DS_D of decision D be able to provide a decision outcome $o \in O_D$ to decision activity $d \in A_d$ that called upon the decision service DS_D.

In summation, a process can call upon a decision by providing a data input set that is required for the enactment of the decision to the decision interface. The interface will steer the information provided by the process towards the suitable decision service. Subsequently, the decision service will invoke the requested decision from the decision model. Consequently, the decision model will enact the decision, reach a decision outcome, and output it to the decision service. The decision service will forward the outcome back to the interface, and through the interface the outcome of the decision will finally reach the process layer. This mechanism is illustrated in Figure 5.2. The SOA design provided in the figure makes it possible to define decisions as services, which we call **Decision as a Service (DaaS)**. These decisions and services can be invoked on demand by information systems, e.g. processes, through a well-defined interface.

With the design provided in Figure 5.2, the SoC paradigm between processes and decisions can easily be respected, as the process is included in the bottom layer, while the decision model is situated in the top layer. Caution is still necessary when modelling the process and decision models: information regarding decisions and decision logic should not be implemented in the bottom process layer, but rather externalised and encapsulated in the decision layer [16, 61, 66, 122].

We link the main elements present in the DaaS example in Figure 5.2 to the formalisation of the DaaS design and DMN constructs as elaborated upon in the current and the previous sections. We use the abbreviations of the names given to elements in Figure 5.2 in the subscript of the symbols to refer to those elements for a compact notation as well as the abbreviations of the data object elements (pertaining to both the process layer and the decision layer) in lower case letters to reference them.

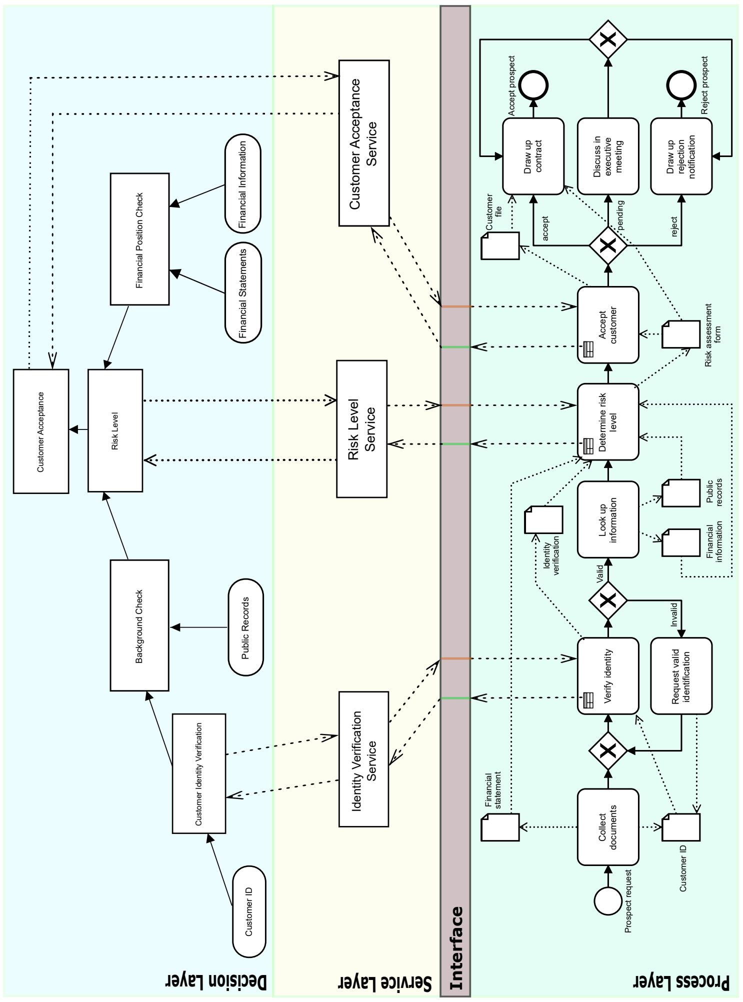


Figure 5.2: Decision as a Service (DaaS) layered architecture.

Decision activities: $A_d = \{a_{VI}, a_{DRL}, a_{AC}\}$.
a_{VI} is a tuple $(cid, iv, L_{D_{CIV}})$;
a_{DRL} is a tuple $(\{fs, iv, fi, pr\}, raf, L_{D_{RL}})$;
a_{AC} is a tuple $(raf, cf, L_{D_{AC}})$.
Decision services: $DS = \{DS_{CIV}, DS_{RL}, DS_{CA}\}$.
DS_{CIV} is a tuple (cid, O_{CIV})
with $s_{CIV} = \{cid\} \in dirs_{CIV}$;
DS_{RL} is a tuple $(\{fs, iv, fi, pr\}, O_{RL})$
with $s_{RL} = \{fs, iv, fi, pr\} \in dirs_{RL}$;
DS_{CA} is a tuple (raf, O_{CA})
with $s_{CA} = \{raf\} \in dirs_{CA}$.
Service interfaces: $IF = \{IF_{CIV}, IF_{RL}, IF_{CA}\}$.
IF_{CIV} is a tuple $(dirs_{CIV}, O_{CIV})$;
IF_{RL} is a tuple $(dirs_{RL}, O_{RL})$;
IF_{CA} is a tuple $(dirs_{CA}, O_{CA})$.
Decisions: $D_{dm} = \{D_{CIV}, D_{BC}, D_{FPC}, D_{RL}, D_{CA}\}$.
D_{CIV} is a tuple (cid, O_{CIV}, L_{CIV});
D_{BC} is a tuple $(\{o_{CIV}, pr\}, O_{BC}, L_{BC})$
with $o_{CIV} \in O_{CIV}$;
D_{FPC} is a tuple $(\{fs, fi\}, O_{FPC}, L_{FPC})$;
D_{RL} is a tuple $(\{o_{BC}, o_{FPC}\}, O_{RL}, L_{RL})$
with $o_{BC} \in O_{BC}$ and $o_{FPC} \in O_{FPC}$;
D_{CA} is a tuple (o_{RL}, O_{CA}, L_{CA})
with $o_{RL} \in O_{RL}$.

5.6 Compliance with the Principles of SOA

In this section, we will evaluate the adherence of the proposed DaaS design to the key Software-Oriented Architecture characteristics as defined in fundamental literature on SOA design [19, 36, 45, 46, 80, 82, 131], both in theory and through the example proposed in Figure 5.2.

5.6.1 Service selection

A client calling upon a service must be able to select the appropriate service based on information provided at runtime. After the service is executed, the client that called the service must be able to interpret the outcome. In the DaaS design, the services are selected at runtime by the process. A process activity provides the service interface IF_D with the decision reference name of the decision D that it wants to invoke and the relevant input data $s_D \in dirs_D$ for that decision. The interface IF_D will select the decision service DS_D that matches with that signature, i.e., input data and requested decision reference. This service is executed and decision outcome $o \in O_D$ is passed on through the interface IF_D to the invoking process activity for further interpretation. Given the modular hierarchy of the decision model, multiple decision services, i.e., with a different input requirements set, can be defined for the same decision. Which decision service will be selected depends on the data input set provided to the interface. This concept is similar to method overloading, which corresponds to methods having the same name and similar functionality but with a different list of arguments. Calling such an overloaded method (abstract decision service) runs one specific definition of the method (a specific decision service) that is appropriate to the context of the call, which is defined by the input set provided to by the process.

In the example in Figure 5.2, decision activity `Determine risk level` provides the input set $\{fs, iv, fi, pr\}$, i.e. the four input data objects of the activity, to the interface. The interface guides, given the input set and the decision requested by the decision activity, the invocation of the correct decision service, i.e. the DS_{RL}. This leads to the invocation of the `Risk Level` decision in the decision layer. Through the service and the decision service interface, the outcome of the decision will be returned to decision activity `Determine risk level` which will produce the *Risk assessment form* containing the decision outcome.

5.6.2 Standardised service communication and loose coupling

Communication between the invoking context, e.g. a business process, and the services, e.g. decision services tied to a decision model, should be standardised and addressed systematically. Besides, a loose coupling between clients and services is preferred above a tight coupling. In a loose coupling there are a few well-defined dependencies between the different modules, while a tight coupling produces a vast array of dependencies that might not all be observed. Furthermore, in order to provide transparency in design and to avoid issues concerning redundancy, services should, in analogy with service communications, be standardised and systematised. The notion of a service should be unambiguous and coherent. In the DaaS design, a decision service DS_D of a decision D is a tuple (s_D, O_D), with $s_D \in dirs_D$ a set of input data sufficient to invoke D and O_D the set of decision outcomes of D. Thus, all new decision services should be designed according to these well-delineated concepts in order to ensure the correct invocation of those new services by potential clients, since the communication between clients from the process layer and the decision service DS_D of a certain decision D is standardised through a well-defined interface IF_D. The interface is the sole channel of communication between the clients and the services, and thus serves as a loose coupling mechanism between the two. Ergo, the clients are aware of the existence of a service, given that the clients can witness and access the interface IF_D of a decision service DS_D. However, the coupling does not go beyond the simple interface connection and the client's awareness of the existence of a decision service.

In the example in Figure 5.2, the `Risk Level Service` is designed as a tuple $(\{fs, iv, fi, pr\}, O_{RL})$ with $s_{RL} = \{fs, iv, fi, pr\}$ and O_{RL} being the output set of the decision `Risk Level`. Thus, the decision service follows the standard design regarding input requirement and output sets. Furthermore, the communication between a decision activity, e.g. `Determine risk level` and the decision service of `Risk Level Service` is standardised and loosely coupled through the decision service interface IF_{RL}, defined as a tuple $(dirs_{RL}, O_{RL})$. The decision activity can only invoke its underlying decision, i.e. `Risk Level` by providing the input data relevant for the invocation of the decision service, i.e. $s_{RL} \in dirs_{RL}$. Likewise, the decision model will render the outcome of the `Risk Level` decision $o_{RL} \in O_{RL}$ to the invoking decision activity through the interface.

5.6.3 Service location transparency

The invoking context of a service should not be burdened with the knowledge of the service location within the network, yet the client should be able to invoke a service at any time, regardless of the service location. This service location transparency allows the service location to change within the network without impairing service availability to clients. In the DaaS design, the process layer is agnostic about the location of a decision service DS_D. Once a correct data input set $s_D \in dirs_D$ is provided to the interface IF_D, the interface will, given the decision D that the client wants to invoke and the set s_D, discover the location of the relevant decision service DS_D and consequently invoke that service. This all happens without any client in the process layer knowing the location of the invoked decision service within the network. Hence, the decision services can be moved around the network and stored anywhere in the network without compromising the availability and accessibility of a decision service.

In the example in Figure 5.2, the `Risk Level Service` is designed as a tuple $(\{fs, iv, fi, pr\}, O_{RL})$. Thus, given the input requirement set $s_{RL} = \{fs, iv, fi, pr\} \in dirs_{RL}$, the interface will discover the location of decision service DS_{RL}, without the client, i.e. decision activity `Determine risk level` being aware of the location of the `Risk Level Service` DS_{RL}.

5.6.4 Service abstraction

The service abstraction characteristic denotes the proposition that services should be conceived as a black box by the invoking context or the clients. Thus, the clients invoking the service are alleviated from the burden of understanding the underlying decision logic and mechanisms the services pertain to. In the DaaS design, the decision logic is encapsulated in the decision model in the top layer, i.e. the decision layer, given that a decision D was defined as a tuple (I_D, O_D, L), where I_D is a set of input symbols, O_D a set of output symbols and L the decision logic defining the relation between symbols in I_D and symbols in O_D. Hence, the decision logic L is encapsulated in decision nodes in the decision model and the services merely invoke the decisions from the decision model and pass the output, through the interface, to the clients in the process layer. Therefore, the clients are agnostic about the underlying decision logic and experience the decision services as a black box. All the clients need to worry about is providing the relevant inputs to the interface of the decision services in order to ensure a sound decision enactment.

In the example in Figure 5.2, the decision logic invoked by the `Determine risk level` decision activity (a_{DRL}) through DS_{RL} is encapsulated in the decision node `Risk Level` (D_{RL}) of the DRD in the decision layer. Hence, the `Determine risk level` decision activity is agnostic about the decision logic underpinning its execution, and the decision activity conceives the `Risk Level Service` DS_{RL}, and consequently the `Risk Level` decision, as a black box that merely returns the decision outcome $o \in O_{RL}$ when provided with an input set $s_{RL} \in dirs_{RL}$.

5.6.5 Service statelessness

Services should be implemented as simple mechanisms that either return the relevant outcome or throw an exception if no conclusion can be reached, i.e. services should be stateless and consequently resource efficient. Thus, the state of the service should be separated from the service itself, providing the service with more flexibility and reusability. In the DaaS design, a decision service DS_D of a decision D is defined as a tuple (s_D, O_D), where $s_D \in dirs_D$ is a set of sufficient input data and O_D the output set representing the decision outcomes of D. Hence, decision services in the DaaS design will return an outcome from the set O_D or an empty set serving as an exception if no conclusion can be reached within a reasonable time frame.

In the example in Figure 5.2, the `Risk Level Service` DS_{RL} simply invokes the `Risk Level` decision at the request of the `Determine risk level` decision activity, and passes the decision outcome outcome $o \in O_{RL}$ back to the invoking decision activity. The `Risk Level Service` itself does not keep a state, but simply serves as a bridge between the process and the decision.

5.6.6 Service longevity

Service longevity denotes the objective that a service should remain unchanged and in existence during a considerably long time. Services should change only when utterly necessary in order to avoid that clients must adapt time and again to the newly changed service. Hence, services should be designed carefully with the intent to last long. In the DaaS design, it is clear that a decision service DS_D of a decision D will only change if the underlying decision model in the decision layer changes as well. Since a decision service DS_D of a decision D is defined as a tuple (s_D, O_D), the decision service DS_D will only need to change if the input set s_D and/or the output set O_D of the underlying decision D are subject to change. Once changes to a decision service occur, clients in the process may need to adapt to ensure a proper invocation of the underlying decisions. Note that if the input sets and output sets of the decision services remain the same, while the underlying decision logic that relates the inputs to the outputs undergoes adaptation, the decision services remain the same from the perspective of the clients, i.e., the processes, since the processes experience the decision logic as a black box.

In the example in Figure 5.2, the `Risk Level Service` DS_{RL} is designed as a tuple $(\{fs, iv, fi, pr\}, O_{RL})$ with $s_{RL} = \{fs, iv, fi, pr\}$ and O_{RL} being the output set of the decision `Risk Level`. Hence, the defined service will only need to undergo change if s_{RL} and/or O_{RL} exhibit any changes. In all other changes to either the decision layer or the process layer, the `Risk Level Service` DS_{RL} remains valid.

5.6.7 Service reusability

Service reusability refers to designing services so that their solution logic is independent of any particular business process or technology, thus eliminating the need for creating new services with the same functionalities for every individual business process that they are required in. That way, the services are not embedded in a single business process, and they can be invoked by different processes and applications as well. Additionally, services should be constructed in a modular way in order to enhance the reuse of parts of the underlying logic. In the DaaS design, the logic L is stored in the decision model as part of the decision layer across multiple decision nodes D, given that a decision D was defined as a tuple (I_D, O_D, L) with the decision logic L being part of the decision D. Ergo, the decision logic is stored in modules represented by the decision nodes, and is not linked to or embedded in any particular business process. For every decision node D multiple decision services DS_D can be created, as discussed in Section 5.5. Hence, the underlying decision logic L can be accessed in smaller modules linked to a number of decision services DS_D, allowing for the invocation and reusability of the decision services by multiple processes.

In the example in Figure 5.2, the logic contained in the `Risk Level` decision node in the decision model can be reused by multiple clients as long as the clients provide the necessary input requirement set $s_{RL} = \{fs, iv, fi, pr\}$ to the `Risk Level Service` DS_{RL}.

5.6.8 Service composability

Several services might be composed into a single larger service thanks to the modular design inherent to the SOA paradigm. This modular design allows to assemble smaller services into coherent larger ones or even into an application. This characteristic is closely related to the characteristic of service granularity, or more specifically: the degree of modularity that the services should have. Decision service composability is inherently enabled by the modular and hierarchical structure of the DMN model, i.e., a higher-level decision is composed of its lower-level decisions. As such, the higher-level decision service is composed of these lower-level decision services as well. Note that composing decision services happens in the decision service layer by combining elements from the decision model. As such, decision service composition is process layer agnostic.

In the DaaS design example in Figure 5.2, multiple decision services or modules are implicitly part of larger decision services or modules. Take for instance the `Risk Level` decision node in the decision layer: for this decision, multiple decision services can be defined. Previously, we defined the `Risk Level Service` DS_{RL} as $(\{fs, iv, fi, pr\}, O_{RL})$ with $s_{RL} = \{fs, iv, fi, pr\}$ and O_{RL} being the output set of the decision `Risk Level`. Another Risk Level Service can be defined as well: DS_{RL_2} as a tuple $(\{o_{BC}, o_{FPC}\}, O_{RL})$ where $o_{BC} \in O_{BC}$ and $o_{FPC} \in O_{FPC}$ are the outputs of the `Background Check` decision and the `Financial Position Check` decision respectively. For both the `Background`

`Check` and the `Financial Position Check` decision, invocable decision services can be defined as well, e.g. DS_{BC} as a tuple $(\{o_{CIV}, pr\}, O_{BC})$ and DS_{FPC} as a tuple $(\{fs, fi\}, O_{FPC})$. Hence, it can be argued that the `Risk Level Service` DS_{RL_2} is a composition of the `Background Check Service` DS_{BC}, the `Financial Position Check Service` DS_{FPC}, and some additional decision logic encapsulated in the module of the `Risk Level` decision node.

5.7 Implications of DaaS for Processes and Decisions

The advantages of the separation of modelling and mining concerns are emphasised in literature [12, 16, 52, 61, 66, 122]. These works especially highlight the scalability, maintainability, flexibility, and understandability of decisions and processes. However, they do not provide a clear design or framework on how to systematically address these issues and how to guarantee a sound SoC by design. In this section we will discuss these advantages of separating the concerns and we will relate them to the DaaS design.

5.7.1 Scalability

A straightforward advantage of separating the process and decision modelling concerns is scalability. Here, scalability does not refer to the performance of a service (scaling with respect to the number of service clients or with the size of service input). Rather, scalability in this chapter refers to the expansion of business process models when adding decision constructs to them. Given the fact that, when separating concerns, the decision logic is not modelled within the process, the decision logic does not clutter the process model and the decision model can be concurrently invoked by many clients. This also promotes the reusability of the decisions and the underlying decision logic.

In the DaaS design in Figure 5.2, it is clear that multiple processes from the process layer can call upon a number of services simultaneously and that all the services can access the decision model at the same time. In cases where the modelling concerns are not separated, e.g. a typical way of modelling the process and decision concerns in one model is using an intricate setup of gateways, the decisions and the decision logic are embedded in the process. This leads to large processes and provides no opportunities for the reuse of logic and decisions and no possibilities for parallel invocation and enactment of the decisions. The DaaS design in Figure 5.2 clearly avoids these convoluted situations, as the decision logic is stored in the decision model as part of the decision layer, while the control flow is part of the process layer. The two do not convolute each other, however, the decisions can easily be invoked by any process as long as the process is able to provide the right input set to the interface of the decision service. That way, the process is not cluttered with decision constructs and multiple processes can access the decision model and make use of the decision logic concurrently.

5.7.2 Maintainability

Another clear advantage of SoC is maintainability. Take for example the situation of a convoluted process-decision model where decisions are hard coded into gateways and the control flow of the process: if either the process or a decision changes, the convoluted model needs to be adapted. This creates a need for constant maintenance of the convoluted process-decision model. When adopting the SoC paradigm, this issues of maintainability and convolution of models are circumvented.

In the DaaS design in Figure 5.2, the decisions and the processes are not convoluted as they are separated into different modules of their respective layers. If parts of a process change in the process layer, the decisions in the decision layer are not affected. This corresponds to a decision-first approach, as opposed to the process-first approach that is present a convoluted process-decision model with cascading gateways [67]. The maintenance of the process does not affect the underlying decisions. On the other hand, if parts of the decision model change, some services and processes will need to adapt to the changes in order to be able to invoke said decisions correctly. However, not all the services and processes will need to undergo adaptations; only the ones that are directly affected by the decisions that were modified. Other services and processes that do not pertain to the adjusted decision will still function properly without any need for adaptation. Noteworthy is that in the case of the DaaS design, where the modelling concerns are separated, all the decision logic is concentrated in the decision model as part of the decision layer. If anything concerning the decisions needs to change, the adaptation happens only in one place, i.e. the decision model. However, if decisions change in the process-first approach, every process containing those decisions will need to adapt as well [13]. Hence, DaaS advances the maintainability of both processes and decisions.

5.7.3 Flexibility

Flexibility refers to the ad-hoc reuse of decisions and decision logic, as well as the flexibility in adapting and changing the underlying decision logic. Clearly, the process-first approach where modelling concerns are not separated does not support flexibility in any way: no reuse of logic is possible, since the logic is embedded in the control flow of the process and not stored in a separate module; and adapting the decision logic is rather cumbersome, since the logic is dispersed across multiple processes and hidden in convoluted process paths. Therefore, the process-first approach shows little flexibility in general.

In the DaaS design in Figure 5.2, the concerns are separated into their respective layers, and as explained earlier, the separated decision logic can be invoked ad-hoc by any client from the process layer conforming to the interface needed to invoke a decision service and consequently the necessary decision. Besides, changing the underlying decision logic is less of a burden, since the logic is concentrated in a single model, rather than dispersed across a process or even across multiple distributed or collaborating processes. Hence, the DaaS design offers a higher level of flexibility, both in terms of reusability of decisions and decision logic, as well as in terms of flexibility in logic adaptation.

5.7.4 Complexity and understandability

The complexity and understandability of models is of particular importance. When modelling concerns are not separated the process quickly becomes overly complicated due to the cascading gateways. This is especially the case in knowledge-intensive processes where a lot of decisions need to be made based on a certain underlying logic. Often the term *spaghetti-like processes* [53] is used to refer to this phenomenon of intricate and convoluted control flows. Furthermore, more understandable models contribute to more maintainable models as well [23].

When opting for a DaaS design, the decisions and the decision logic are externalised and encapsulated in a separate layer as part of the decision model. Therefore, the control flow of the process is alleviated from the burden of representing the different decision paths. As a consequence, the actual process becomes visible and more understandable. However, because of the fact that decisions have been externalised, the process is still burdened with data management and data propagation, i.e. the process is responsible for the collection and propagation of the data needed for the invocation of a decision service. The decision model does not concern itself with these issues of data propagation and data management. It simply transforms the input data, obtained from the process through the decision service, into a decision output which is sent back to the invoking process. Hence, in the DaaS design, process complexity in terms of control flow will decrease, however, the complexity of data management is likely to increase. Thus data management and data propagation within processes becomes of paramount importance. Besides, when separating the decisions from the process, the overall view of the entire problem might be clouded, as processes take abstraction of decisions and simply approach decisions as a black box that answers to input, without concerning themselves with the underlying decision logic. Therefore, when applying the DaaS design, the emphasis should be put on the decisions and on a decision-first approach. The process might take abstractions from decisions and conceive them as a black box, however, everything stands or falls with the correct definition of the decisions, as both the decision services and the processes need to heavily rely on the decisions to function properly.

5.8 Evaluation of DaaS Design on a Real-Life Event Log

In this section, the DaaS approach is illustrated with automatically discovered decision services from an enriched event log. Literature on automatic discovery of DMN decision models from event logs has seen a considerable surge over the past few years [12, 32, 122]. The work in [122] is particularly interesting as it addresses decisions within a processes over multiple activities or even across the entire process execution span, rather than containing decisions to local decision points in the process. This **P**rocess **M**ining **In**tegrating **D**ecisions **(P-MInD)** framework provides interesting insights in the interaction between processes on the one hand, and data, rules, and decisions on the other. We capitalise on the findings in [122] by adapting the technique in order to unveil decision services within processes, which will better explain the interaction between the processes and the decisions. In the remainder of this section we explain the changes applied to P-MInD needed to acknowledge decision services, thus rendering the discovered models consistent with the SOA paradigm. We call this approach the **S**ervice-**O**riented **A**rchitecture **P**rocess **M**ining **In**tegrating **D**ecisions **(SOAP-MInD)**. This section is concluded with a real-world example of DaaS-compatible decision services derived from an enriched event log containing information on a bank loan application and approval process.

5.8.1 SOAP-MInD

While P-MInD [122] discovers holistic decision models from event logs, it does not clearly illustrate how the process model communicates with its underlying decision model. In order to clarify the decision invocations by the process it is necessary to explain which decisions the process invokes at which specific points in the process. Additionally, it is required to know how the process invokes the desired decision, i.e. with which input data objects. In other words, for each invoked decision D the decision services DS_D that are called upon by the process need to be identified. P-MInD abstracts from the idea of services and is only concerned with the decision layer and process layer in

Figure 5.2. However, to understand the interactions between the process and decision layers, the specific invocations of the process layer, i.e. the service calls, need to be acknowledged. Thus, we expand P-MInD to **S**ervice-**O**riented **A**rchitecture **P**rocess **M**ining **In**tegrating **D**ecisions **(SOAP-MInD)**, a framework for mining integrated decision services and the respective process traces where they are invoked. SOAP-MInD builds further on the principles of P-MInD and we refer to that seminal work on holistic decision modelling for details [122]. In what follows, we briefly describe the adaptation of P-MInD to render SOAP-MInD. SOAP-MInD was made compatible with the ProM framework as it is available as a ProM plugin, thus making the code open source[7].

The basis for both the P-MInD and SOAP-MInD approaches lies in the classification of activities according to Definition 5.8. This classification aides in understanding the interaction between the decision services and the decision service interface on the one hand, and the process itself on the other. Clearly, the set of operational activities (A_o) is not involved in decision making and is therefore irrelevant for the decision service layer and the decision service interface. Administrative activities have the purpose to introduce input data objects for decisions and are therefore relevant for the decision service layer and interface. An administrative activity $a \in A_a$ will provide an output set $O(a)$ which can be used as inputs for future decision activities in the process. A decision activity $d \in A_d$ will receive an input set $I(d)$ which is composed of the outputs of previously executed administrative activities and/or outputs of previously executed decision activities. The decision activity $d \in A_d$ will call upon the decision service DS_D through the decision service interface IF_D in order to invoke underlying decision D.

Algorithm 1 captures how both the the decisions are stored and possible decision services are identified. First, the influence of activities over variables is identified through checking which shifts exist in an event log, i.e., whether the value of a variable v changed during an activity a compared to its previous occurrence in a trace. This is taken as evidence that the activity influences v, and both are stored as potential decisions (lines 2-3). Next, all potential decisions are checked for their occurrence, to see whether

[7]`https://svn.win.tue.nl/repos/prom/Packages/PMinD/`

a certain decision happens (enough times, see *mintraces*) before another ($a_i < a_j$), and hence whether it might have an influence over each other's variables (lines 4-5). Note that this allows for decisions in which a_i serves as input to a_j, and vice versa. For all traces where this is the case, the correlation between both variables is calculated to see whether there is a link between both decisions (line 6). The variable/activity pair (VP) occurring the latest ((v_j, a_j)) is stored as a potential decision with inputs and outputs including the variable(s) of the earlier occurring activity a_i. This happens until all these connections are made, and the full input set of a_j is established. Next, all different decision sequences that happen in all traces are distilled, depending on what decision-relations were established through the correlations. After this, all possible decision services are discovered as well, in set DS. For each cluster of decisions, the values attached to the corresponding variable/activity pair are used to train a predictive model L_d, completing the decision (lines 10-13). More details on the extraction of the shifts, as well as correlation and thresholds are discussed in [122]. Furthermore, due to page restrictions, we provide a parameter assessment of SOAP-MInD in a technical report published at our institution [68].

Algorithm 1 Decision service discovery in an event log.

```
1: procedure FIND_DECISION_SERVICES(event log $\mathcal{L}$)
2:     Retrieve all shifts in the event log, i.e., all activities $a \in A$ for which a variable's
       $v$ value changes during its execution
3:     $VP \leftarrow (v, a)$, a potential candidate decision $D$ where $v \in O_{d^a}$ and $a \in A_d$
4:     for $(v_i, a_i), (v_j, a_j) \in VP \times VP$ do
5:         if $|\{t \in \mathcal{L} \mid a_i, a_j \in t \wedge a_i < a_j \wedge v_i \in I(v_j)\}| > mintraces$ then
6:             if $corr(v_i, v_j) > corrthres$ then
7:                 $D \leftarrow d = (I_d \leftarrow v_i, O_d = v_j, L = \emptyset)$        ▷ See Def. 5.2
8:                 $DS_d \in DS \leftarrow (I_d, O_d)$
9:             end if
10:        end if
11:    end for
12:    Cluster traces in set $C_T$ where the same subsets $DS_{C_T} \subseteq DS_d$ are present
13:    for $c \in C_T$ do
14:        for $DS_d \in DS_{C_T}$ do
15:            train predictive model $L_d$ over $I_d$ to predict $O_d$
16:            $D \leftarrow d = (I_d, O_d, L_d)$
17:        end for
18:    end for
19: end procedure
```

Hence, the trace clusters are grouped based on the decision services they invoke. Note that in Definition 5.6, a decision service DS_d of a decision d was defined as a tuple (s_d, O_d), where $s_d \in dirs_d$ is a set of input data sufficient to invoke the decision d and O_d the output set representing the decision outcomes of d. The traces in this step of the SOAP-MInD approach are clustered based on the set of input data $s_d \in dirs_d$ needed to invoke a decision d and to obtain a decision outcome $o \in O_d$. Hence, the clusters present different execution sequences in which the decision services were invoked through the interfaces.

5.8.2 Decision service compliance verification

SOAP-MInD renders two models per trace cluster of variable shift sequences: a process model and a decision service model. This offers opportunities of decision service compliance verification, i.e. per trace cluster it can be investigated whether the discovered process model is able to correctly invoke the services pertaining to that same trace cluster.

Naively it can be stated that the process model either complies with the decision service or it does not comply with the service, as defined in the decision **Service Adherence Criterion (SAC)** in Definition 5.10. If the process model complies with the decision service, it inherently provides the necessary decision input requirements set $s_D \in dirs_D$ needed for the invocation of decision service DS_D pertaining to decision D. Clearly, this input requirements set s_D must be readily available to the process before the point in the process where the decision service DS_D is invoked. That way, the process can correctly provide the decision service interface IF_D with its corresponding input data. Consequently, the decision service DS_D will invoke decision D and an outcome $o \in O_D$ will be returned to the process through the interface IF_D. However, if no valid decision input requirements set $s_D \in dirs_D$ is available in the process at the time when the process invokes decision D by calling upon its decision service DS_D through the interface IF_D, the process is not complying with the decision service. Hence, no crisp decision outcome $o \in O_D$ of decision D will be returned to the process by the decision model through the decision service DS_D and the decision interface IF_D.

This is a rather naive approach towards decision service compliance verification, as decision service invocations are only considered to enact an underlying decision if the input data provided is both complete and correct. However, decision reasoning on incorrect and incomplete data can be applied as well. If for instance one input data element is missing from the decision input requirements set $s_D \in dirs_D$, the decision service DS_D might still be able to invoke its underlying decision model and provide all possible decision outcomes, given the known values is s_D and all possible values for the missing data element in s_D. The process would then, instead of one decision outcome, be provided with a set of possible decision outcomes $O_{D_M} \in O_D$. As a result, the process could still be able to continue properly after the decision enactment and reach a sound conclusion, given the correct interpretation of the decision outcomes in the process flow. Hence, the situation is more nuanced as it can be argued that a process can partially comply with a decision service as well, i.e. the process does not fully comply with the decision service in terms of input requirements sets, data hierarchies, and data propagation, however, given the data that the process provides to the decision service, a set of decision outcomes might still be reachable and useful for further process enactment. In that case, the decision Service Adherence Criterion (SAC) of Definition 5.10 can be relaxed towards a weaker version of decision service compliance.

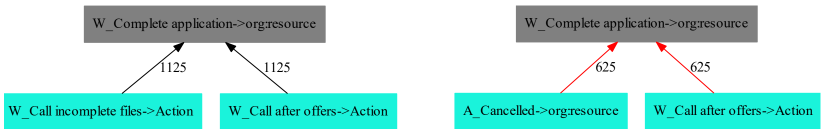

W_Complete application->org:resource
1125
1125
W_Call incomplete files->Action
W_Call after offers->Action
W_Complete application->org:resource
625
625
A_Cancelled->org:resource
W_Call after offers->Action

Figure 5.3: Discovered decision services.

5.8.3 Illustration and discussion

In this subsection, we illustrate the DaaS design by applying the SOAP-MInD framework to a real-life enriched event log containing information on a bank loan application and approval process, made available for the 2017 BPI Challenge[8]. The log was filtered to a time

[8]`https://doi.org/10.4121/uuid:5f3067df-f10b-45da-b98b-86ae4c7a310b`

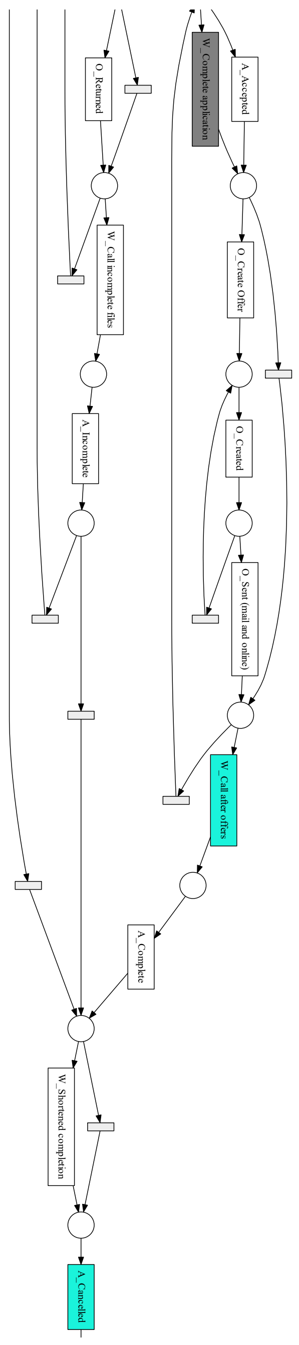


Figure 5.4: Fragment of the trace cluster not adhering to the decision service.

window containing data between 26 August 2016 and 28 October 2016, i.e. 9 weeks of data. The filtered log contains 77,317 events over 4,382 cases. No additional pre-processing was performed and the unaltered log was used to create the output. Figure 5.3 provides two decision services that were extracted from the log. The decision services pertain to two trace clusters. Figure 5.4 depicts a part of the mined trace cluster relating to the rightmost decision service from Figure 5.4. Due to page restrictions, the full mined process models representing the trace clusters have been made available online[9,10].

The leftmost decision service in Figure 5.3 is invoked in the trace cluster containing 1,125 traces. In these traces, the `Complete application` decision has to be made. In order for the service to invoke this decision, the outcomes of two subdecisions need to be provided by the process to the decision service. The outcomes of the `Call incomplete files` and `Call after offers` decisions form the input requirement set needed for the correct invocation of the `Complete application` decision. Hence, this DMN model represents a decision service that is invoked in the trace cluster it pertains to. Note that in this model the information requirement arrows are coloured black, indicating that the trace cluster conforms to the discovered decision service, i.e. the **Service Adherence Criterion (SAC)** in Definition 5.10 is adhered to in terms of input requirements for the decisions that are invoked. Thus, the order of the decision activities in the process model conforms to the hierarchy as depicted in the decision service model.

On the other hand, the rightmost decision service in Figure 5.3 is not adhered to by the trace cluster which invokes that decision service. Notice that the arrows in this model are coloured red to indicate that the order of decision activities in the process model, representing the 625 traces cluster, violates the decision hierarchy demanded by the decision service model. Namely, the service requires the outputs of the `Application cancelled` and `Call after offers` decisions in order to invoke the `Complete application` decision. As shown in the corresponding process model in Figure 5.4, and the full process model that is provided

[9]`https://feb.kuleuven.be/public/u0111379/TSC/`

[10]The mined trace cluster relating to the leftmost decision service from Figure 5.4 is provided in Figure B.1 in Appendix B.

online, this hierarchy is not respected and at the time of invocation of the decision service in the process, the necessary input requirement set is not available to the process. As a consequence, the process can not call upon the decision service in a proper manner and hence the decision model can not enact the invoked decision without the relevant input data. Thus, no sound decision outcome will be provided to the process, and the process might not be able to resume.

Next to the input requirement arrows in Figure 5.3 the number of traces that invoke the decision service are depicted. These traces are clustered according to the decision input and output propagation, as discussed in Section 5.8.1. Hence, the data propagation within the trace indicates which input requirement sets are passed on within the process and thus which decision services are invoked at specific points in the process. Thus, service-orientation and SoC provide a view into the intersection and interplay between data and processes, process modelling and decision modelling, process mining and classical data mining. Capitalising on the DaaS design, the advantages inherent to service-orientation can be exploited when separating multi-perspective modelling and mining tasks, such as the process perspective and the decision perspective.

5.9 Evaluating DaaS SOA Maturity

Maturity models can be used to assess the maturity of a system and thus to provide a roadmap towards a successful implementation of the system. For the purpose of SOA adoption, SOA maturity models have been proposed in literature. However, [110] point out that existing SOA maturity models are in most cases developed by vendors of SOA solutions and that they are therefore dependent on the specific products they are designed for. Hence, they propose an independent SOA Maturity Model (iSOAMM), i.e., a SOA maturity model that is independent of the used technologies and products. They develop SOA maturity model levels which are oriented at the capability of an SOA to support business processes. This means that a SOA with higher maturity possesses more features, which are useful within business processes. An overview of the iSOAMM maturity levels for the architectural viewpoint is provided in Figure 5.5.

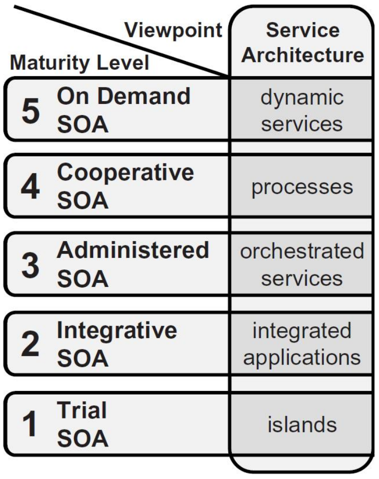


Figure 5.5: Independent SOA Maturity Model (iSOAMM) [110]

Level 1: Trial SOA: This level recognises the existence of services. However, different services use incompatible technologies and standards. Hence, the services exhibit a lack of standardisation and they form a collection of unconnected service islands and not a true service-oriented architecture.

Level 2: Integrative SOA: This maturity level introduces a standardised service interface such that a high-level application, e.g., a business process, can use the interface to access the different services that are provided by the system.

Level 3: Administered SOA: This maturity level introduces service orchestration, i.e., it allows for composing several existing fine-grained services into a single higher order composite service. This modular service approach promotes the reuse and manageability of service components.

Level 4: Cooperative SOA: This level supports the choreography of processes, i.e., cooperation between processes. Additionally, human users are often vital to process execution support. Hence, choreography can be employed to close the gap between services on the one hand, and human users and (external) business processes on the other.

Level 5: On Demand SOA: In this maturity level, the predefined services are replaced with service selection at runtime.

In what follows, we briefly explain the adherence to each maturity level and we apply the iSOAMM to the DaaS design proposed in this chapter. The DaaS design conforms to maturity level 1 as it provides services. Since the design provides a standardised service interface in the form of Definition 5.7, and since services are developed according to Definition 5.6, using the same standard, i.e., DMN, the DaaS design conforms to maturity level 2 as well. Given the modular design of DMN decision models, as defined in Definitions 5.1 and 5.3, the designed decision services exhibit modularity as well as they provide opportunities for decision service composition and decomposition as explained in Subsection 5.6.8. This ensures conformance to maturity level 3. Adherence to maturity level 4, i.e., cooperation between processes and human user support of processes, is not explicitly included in the DaaS design, as collaborative business processes were considered out of scope in this chapter. However, the DaaS design allows for adding another layer that is concerned with communication between the process in the DaaS design and human actors or external processes. Furthermore, the process layer inherently allows for choreography between process and human users by employing pools and lanes within the process models. Finally, the DaaS design adheres to maturity level 5 as well, since it provides a decision on demand service to business processes, i.e., the services are invoked at runtime by the process by providing the decision name that the process wants to invoke, together with the input set necessary for the invocation of the decision.

5.10 Limitations of the DaaS Design

Organising process-aware information systems (PAIS) according to the SOA DaaS design requires system redesign. This induces limitations and obstacles as the processes in the legacy system did not follow the SOA and SoC paradigms. First, the decisions embedded in the process flows need to be externalised into a separate DMN decision model. This leads to the introduction of additional complexity because of new types of models that are becoming part of the system, i.e., the decision models. Furthermore, the business processes need to undergo redesign [67] to eliminate the decision construct that were present in the legacy process models. This is done to avoid ambiguity and overlap between decision constructs in the process model and the decision specified in the decision model. As such, the SoC paradigm between processes and decisions is instated. Additionally, the redesigned process models need to be compatible with the newly established decision models and hence the decision services derived from them. Thus, decision service consistency constraints apply according to the service adherence criterion when designing processes that need to call upon the decision logic through the decision services. Hence, a more complicated data propagation management within processes is inherent to the DaaS design, as opposed to designs where the SoC paradigm is not respected.

5.11 Conclusion

In this chapter we have contributed a *SOA* design for process- and decision-aware information systems, enhancing the understanding of the interaction between decisions and processes according to the SoC paradigm. The provided framework consists of a process layer, interface, service layer, and decision layer, making possible the implementation of decisions as services, or **Decision as a Service (DaaS)**. The processes can access the decisions through this DaaS architecture on demand, which we named **Decision on Demand (DoD)**. Furthermore, this chapter formally defines the key concepts of DaaS and DoD and the proposed design is evaluated against key SOA characteristics, elucidating the benefits in terms of service abstraction and usefulness of the DaaS/DoD mechanism. Furthermore, implications of the proposed framework regarding integrated process-decision modelling are elaborated upon as well, demonstrating that the DaaS design greatly benefits the SoC between processes and decisions, thus advancing scalability, maintainability, flexibility and understandability. Additionally, we illustrated that the proposed DaaS design exhibits itself in real-life event logs by applying automatic decision service discovery on an enriched event log on a real-life bank loan application and approval process. Finally, the DaaS design was assessed in terms of SOA maturity, illustrating that the design conforms to the highest maturity levels.

CHAPTER 6

Parameter Assessment of the Automated Decision Service Discovery

"The goal to be reached is the mind's insight into what knowing is. Impatience asks for the impossible, wants to reach the goal without the means of getting there. The length of the journey has to be borne with, for every moment is necessary; [...] because by nothing less could that all-pervading mind ever manage to become conscious of what itself is — for that reason, the individual mind [...] cannot expect by less toil to grasp what its own substance contains."

The Phenomenology of Spirit
— *Georg Wilhelm Friedrich Hegel*

This chapter presents sections 4 and 5 of the following paper:

Faruk Hasić, Johannes De Smedt, Seppe vanden Broucke, Estefanía Serral Asensio. A Parameter Assessment of Service-Oriented Architecture Process Mining Integrating Decisions (SOAP-MInD). `FEB Research Report KBI_1914 (KU Leuven)`, Leuven (Belgium), 1-9, 2019.

Abstract. This chapter provides a parameter assessment of the Service-Oriented Architecture Process Mining Integrating Decisions (SOAP-MInD) technique introduced in the previous chapter. For this purpose, we capitalise on the existing Process Mining Integrating Decisions (P-MInD) framework, which allows decisions to be mined from a process event log. We adapted P-MInD to the Service-Oriented Architecture (SOA) paradigm in order to reflect the interaction between processes and decisions in trace clusters that are constructed based on the sequence of input data interactions between the process and the decisions. The code for this adapted Service-Oriented Architecture Process Mining Integrating Decisions (SOAP-MInD) is provided as an open-source plugin in the ProM framework for process mining. We illustrate the workings of SOAP-MInD by running a parameter assessment and providing the results in terms of the amount of discovered trace clusters, decision services, the granularity of the decision services, and the run time.

6.1 Parameter Assessment

To illustrate the existence of decision logic invocations in real-life enriched event logs, we apply the SOAP-MInD framework on a bank loan application and approval event log, made available for the 2017 Business Process Intelligence (BPI) Challenge[11]. The event log consists of 1,202,267 events in 31,509 traces, containing 26 activity types and 18 variables.

Table 6.1 provides the assessment of run experiments with different parameter levels and information regarding the models obtained with these parameter settings. All the parameters are expressed in percentages. The parameter *minsup* is used to establish whether there was a causal link between a shifted variable and the other variables in the process activity. To determine whether the activity alters the value of a particular variable enough times to assume there is an influence over the variable, the parameter *shiftratio* is established. The parameter *corrthres* determines whether to store the inputs that are correlated with values of a variable. Depending on the overlap of the traces, new trace clusters are made. This is managed by the parameter *mindev*, which ensures a sufficient enough deviation in clusters before splitting them up into separate ones. Finally, the parameter *mintrace* is used to see whether a certain decision happens enough times before another, and hence whether it might have an influence over each other's variables. For a more detailed account of the parameters, we refer to [122].

For every combination of the input parameters, Table 6.1 provides the time (in milliseconds) it took ProM to run on a desktop (Xeon E3-1230 v5 CPU, 32GB RAM in Java 8) to obtain the decision requirement diagram (DRD) models. Furthermore, the number of decision models, which also corresponds to the number of trace clusters that were discovered, is recorded. Additionally, the average number of nodes in the DRD, the average number of edges in the DRD, the average DRD depth, and the maximum DRD depth are displayed. These are all taken across all the discovered DRDs in all the trace clusters pertaining to a parameter combination. It is clear from Table 6.1 that the longest running

[11]`https://doi.org/10.4121/uuid:5f3067df-f10b-45da-b98b-86ae4c7a310b`

Table 6.1: Parameter assessment applied to the BPI 2017 Challenge log. Parameters are expressed in percentages.

minsup	**shiftratio**	**corrthres**	**mindev**	**mintrace**	**run time**	**#models**	**#nodes**	**#edges**	**depth**	**maxdepth**
60	20	10	80	20	961967	19	4	5	2	4
				30	803001	9	3	3	2	3
			90	20	1037709	19	4	5	2	4
				30	890162	9	3	3	2	3
		20	80	20	865582	12	3	2	2	3
				30	779830	8	3	2	2	3
			90	20	923489	12	3	2	2	3
				30	839108	8	3	2	2	3
		30	80	20	821627	7	3	2	2	3
				30	754334	5	3	2	2	3
			90	20	799169	7	3	2	2	3
				30	765590	5	3	2	2	3
	30	10	80	20	770324	16	5	5	2	4
				30	612680	7	4	4	2	3
			90	20	722334	16	5	5	2	4
				30	593650	7	4	4	2	3
		20	80	20	629718	10	3	2	2	3
				30	584915	6	3	2	2	3
			90	20	618483	10	3	2	2	3
				30	568513	6	3	2	2	3
		30	80	20	597231	5	3	2	2	3
				30	611252	3	3	2	2	3
			90	20	663545	5	3	2	2	3
				30	597269	3	3	2	2	3
70	20	10	80	20	906217	19	4	5	2	4
				30	746997	9	3	3	2	3
			90	20	830106	19	4	5	2	4
				30	743171	9	3	3	2	3
		20	80	20	745552	12	3	2	2	3
				30	707089	8	3	2	2	3
			90	20	736886	12	3	2	2	3
				30	686325	8	3	2	2	3
		30	80	20	704901	7	3	2	2	3
				30	684567	5	3	2	2	3
			90	20	735997	7	3	2	2	3
				30	672633	5	3	2	2	3
	30	10	80	20	814774	16	5	5	2	4
				30	707149	7	4	4	2	3
			90	20	808342	16	5	5	2	4
				30	690853	7	4	4	2	3
		20	80	20	720558	10	3	2	2	3
				30	662769	6	3	2	2	3
			90	20	716551	10	3	2	2	3
				30	693305	6	3	2	2	3
		30	80	20	703106	5	3	2	2	3
				30	649494	3	3	2	2	3
			90	20	677885	5	3	2	2	3
				30	655080	3	3	2	2	3

parameter combination runs for 1,037,709 milliseconds and that the average number of nodes and the average number of edges per model never exceed five, thus rendering models that are relatively small and comprehensible. All the unique decision services that were discovered for the parameter settings in Table 6.1 are provided online in the form of DMN decision requirement diagrams in Portable Network Graphics (PNG) format[12].

6.2 Conclusion

In this chapter, we have performed a parameter assessment of the SOAP-MInD technique, an adaptation of P-MInD [122] that is capable of representing modular service-oriented decision invocations that can be spread across multiple places within a process. Furthermore, SOAP-MInD shows that different process variants interact with the decisions in a different fashion, since the traces are clustered according to decision variable shift sequences in the process. SOAP-MInD is implemented as a plug-in to the ProM framework for process mining, hence making the source code open-source.

[12]https://feb.kuleuven.be/public/u0111379/TSC/Assessment

Part IV

Change Patterns, Model Evolution, and Tool Support

CHAPTER 7

Decision Model Change Patterns for Dynamic System Evolution

'The chief benefit, which results from philosophy, arises in an indirect manner, and proceeds more from its secret, insensible influence, than from its immediate application."

Treatise of Human Nature
— *David Hume*

This chapter was published as follows:

Faruk Hasić, Carl Corea, Jonas Blatt, Patrick Delfmann, Estefanía Serral Asensio. Decision Model Change Patterns for Dynamic System Evolution. `Knowledge And Information Systems`, *article in press*, 2020.

A preliminary version of this paper was published in:

Faruk Hasić, Estefanía Serral Asensio. Change Patterns for Decision Model and Notation (DMN) Model Evolution. `The 18th Belgium-Netherlands Software Evolution Workshop (BENEVOL)`, Brussels (Belgium), *article in press*, 2020.

Abstract. In the modern digital era, information systems must operate in increasingly interconnected and dynamic environments, which force them to be changeable yet consistent. Such modern information systems are usually decision- and knowledge-intensive. A recently introduced standard, the Decision Model and Notation (DMN), has been adopted in both industry and academia as a suitable method for modelling decisions and decision rules. Noteworthy is that, despite the dynamic nature of modern knowledge-intensive systems, DMN was only studied and implemented in a static fashion, as decision schema change patterns have not received any attention so far. This chapter identifies and analyses the change patterns that can occur in a DMN decision model. A change in the decision model can require the triggering of other changes in order to safeguard consistency. As such, this chapter will also investigate for each change pattern which further changes should be performed to ensure model consistency. The patterns presented in this chapter will not only facilitate the understanding of decision change management and within-model consistency, but can also be capitalised on for developing and implementing flexible decision management systems. To illustrate this, we present a modelling environment prototype that provides modelling support when applying the proposed change patterns.

7.1 Introduction

Decision Model and Notation (DMN) is a recently introduced decision modelling standard that has enjoyed significant interest in literature [24, 25, 42, 58, 59, 66, 76, 106]. 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 subdecisions, ovals to represent data input, corner-cut rectangles for business knowledge models, and curved rectangles to represent knowledge sources. A schematic example of a DRD is provided in Figure 7.1. The decision logic level is usually represented in the form of decision tables.

DMN has mainly been adopted in business process management (BPM) literature, which moves towards accommodating decision management into the paradigms of Separation of Concerns (SoC) and Service-Oriented Architecture (SOA), by externalising decisions and encapsulating them into separate decision models [32, 39, 66, 74, 100, 116]. Hence, the decisions are implemented as externalised services. This externalisation of decisions from processes, provides a plethora of advantages regarding maintainability and flexibility for the process [49, 60, 66, 70, 76, 79]. However, all current works approach DMN from a static perspective, i.e., no attention has been given to changing or adaptable decision models. Nevertheless, the dynamic nature of modern knowledge-intensive systems demands a degree of flexibility to cope with the changing requirements [57]. In this chapter we approach this research gap by identifying and defining the change patterns that can be applied to DMN decision models. Additionally, we asses the effect of each applied change pattern on decision model consistency and we investigate and determine the change propagation that is needed to be triggered to keep such consistency. A change pattern is defined as an operation that inserts, deletes, moves, or modifies model elements [55]. Note that moving or modifying model elements can be achieved by applying the insert and delete patterns, as such the insert and delete patterns form the core change patterns that will be the subject of this chapter. We also refer to them as as inclusion and exclusion patterns. The findings in this chapter can aid in developing and implementing dynamic decision management tools and systems that meet the requirements that the digital era demands. To illustrate this, we also provide a proof-of-concept tool for modelling support.

The contribution of this chapter is fourfold:

1. We identify and define change patterns for DMN decision models.

2. We analyse the influence of each decision model change pattern on the within-model consistency.

3. We investigate the propagation of decision model change patterns throughout the decision model, i.e., we discuss the chain of change patterns that can be set in motion to restore within-model consistency after applying a single change pattern on the decision model.

4. We develop a proof-of-concept modelling environment prototype which provides modelling support when applying the change patterns proposed in this chapter.

This chapter is structured as follows. Section 7.2 discusses related work and Section 7.3 provides preliminary formalisation needed for the construction of the change patterns. In Section 7.4 we introduce a running example of a DMN decision model which will be used to illustrate the change patterns. Section 7.5 introduces a set of DMN decision model change patterns, while Section 7.6 discusses the inconsistencies and change propagation induced by the change patterns. In Section 7.7 we present the modelling environment prototype which provides modelling support for change pattern administration in DMN decision models. Finally, Section 7.8 provides conclusions and directions for future research.

7.2 Related Work

DMN was first introduced by the Object Management Group (OMG) in 2015. Since then, several works on DMN have emerged. Most works have focused on the integration of process and decision models from a modelling point of view e.g., [9, 16, 17, 18, 60, 62, 65, 66, 67]. Others focus on the automatic discovery of decision model from enriched process event logs [32, 123]. Works on soundness of DMN models have been proposed as well [6, 10]. Furthermore, literature provides a set of tools for modelling DMN models [39, 74, 75, 76]. Before the introduction of DMN, The Decision Model (TDM) was introduced for modelling business logic and as a new requirements artifact for decision-aware business processes [138, 139]. Furthermore, works on ontologies for knowledge-intensive business processes have been suggested [43, 117]. These consider additional relevant aspects in decision making which DMN does not cover or which cannot always be adequately represented in a decision table, such as advantages, disadvantages, risks, facts, evidence and feelings. Nonetheless, in certain knowledge-intensive settings, these aspects can cause changes that affect the consistency of a decision model.

The ability of a knowledge-based system to efficiently deal with decision rule changes is considered of paramount importance in literature [15, 26, 41, 86, 102]. This way, business rigidity is avoided and the ability to transform the underlying business rules to new realities is facilitated. However, the lack of research on process models at runtime is emphasised in [128], and 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. Nevertheless, adaptable models are considered of paramount importance [20, 92, 128]. 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 [148]. 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.

Change propagation throughout the model is an important aspect in hierarchical structures where referential integrity needs to be upheld. Typically, such structures are connected by tables, such as the relational database structure. The changes in relational tables are propagated by rules such as on delete cascade or on update cascade [27, 107]. Decision tables that are used in decision models are of a similar hierarchical structure, and hence, similar change propagation can be expected. However, instead of cascading an update or delete action throughout the whole model, the change propagation can also be captured by applying other change patterns. Furthermore, the authors of [11, 124] show that, despite referential integrity being researched for database systems, the verification capabilities for business rules management are still lacking. As such, change patterns for decision models were not addressed yet in literature and, to the best of our knowledge, this is the first work on change pattern propagation for DMN decision models.

7.3 Preliminaries

In this section, we also provide a formalisation for key DMN concepts needed for the development of the change patterns that will be discussed in the following sections. For the formalisation, we rely on [13, 66]. The backbone of a DMN decision model is formed by rectangles that depict decisions and subdecisions and ovals that represent data input. Additionally, business knowledge models and knowledge sources can be defined. The underlying decision logic is usually represented in decision table form.

Definition 7.1 (Decision requirement diagram (DRD))**.** A DRD is a tuple (D_{dm}, ID, BK, KS, IR, KR, AR) consisting of a finite non-empty set of decision nodes D_{dm}, a finite non-empty set of input data nodes ID, a finite non-empty set of business knowledge model nodes BK, and a finite non-empty set of knowledge source nodes KS. These nodes are connected by requirements into a directed acyclic graph (DAG): a finite non-empty set of directed edges IR representing the information requirements such that $IR \subseteq (D_{dm} \cup ID) \times D_{dm}$, a finite non-empty set of knowledge requirements KR such that $KR \subseteq BK \times (D_{dm} \cup BK)$, and a finite non-empty set of authority requirements AR such that $AR \subseteq (D_{dm} \cup ID \cup KS) \times (D_{dm} \cup BK \cup KS)$.

A schematic DRD is represented in Figure 7.1. According to the DMN standard, a decision requirement graph can be an incomplete or partial representation of the decision requirements in a decision model. The set of all DRDs in the decision model constitutes the exhaustive set of requirements. The information contained in this set can be combined into a single DRD representing the decision requirements level as a whole. The DMN standard refers to such a DRD as a decision requirement graph (DRG). We expand the notion of a DRG, in such a way that a DRG is a DRD which is self-contained, i.e. for every decision in the diagram all its requirements are also represented in the diagram.

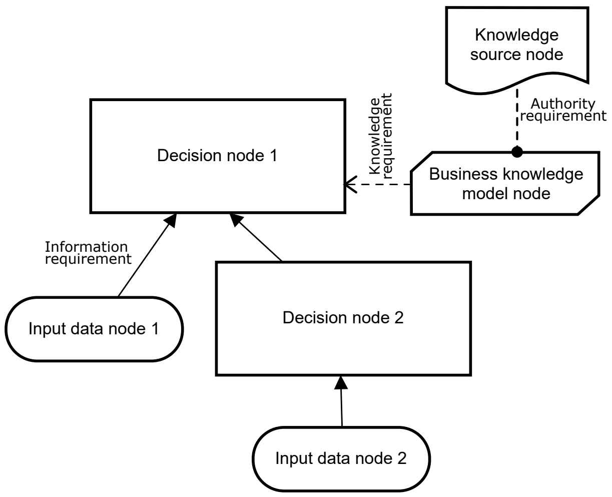


Figure 7.1: A schematic overview of DRD model elements.

Definition 7.2 (DRG)**.** A decision requirement diagram DRD is a decision requirement graph DRG if and only if for every decision in the diagram all its requirements are also represented in the diagram.

The term *decision* can have a number of meanings. According to the DMN specification a decision is the logic used to determine an output from a given input. Meanwhile, in process modelling an activity can be a decision task, e.g., the business rule task in the Business Process Model and Notation (BPMN) standard [105] in which a decision is made by applying decision logic [139].

Another common meaning is that a decision is the actual result, which we call the output of a decision, or simply the decision result. We define a decision as follows:

Definition 7.3 (Decision). A decision $d \in D_{dm}$ is a tuple (I_d, O_d, L), where $I \subseteq ID$ is a set of input symbols, O a set of output or result symbols and L the decision logic defining the relation between symbols in I_d and symbols in O_d.

In case of decision tables, I and O contain the variables of the input and output elements respectively, and L is the table itself, i.e. the set of decision rules present in the table.

In DRDs these decisions d_i are contained by the decision nodes $D_i \in D_{dm}$. One can state that d_i and D_i are equivalent views at different levels of granularity: d_i looks at a decision on its own, while D_i places d_i in the hierarchy of the DRD. We will use D to refer to both a decision and its representing node in a DRD. From the definition of DRGs, it is clear that every decision D in a DMN model has a unique decision requirement graph DRG_D with D as its single top-level decision. A DRG contains all information requirements of its top level decisions. Hence, only one DRG exists with D as its single top-level decision, i.e., DRG_D. Furthermore, all the decisions in a DRG_D, except D, are consequently subdecisions of D. In other words, the top-level decision *requires* these lower level subdecisions.

7.4 Running Example of a DMN Decision Model

Consider a patient health monitoring system for a person diagnosed with the Chronic Obstructive Pulmonary Disease (COPD). COPD is a disease that obstructs the lungs and obstructs the airflow and breathing of the patient. Acute attacks of the disease can happen. In that case the patient can experience uncomfortable complications such as fast breathing, a fast heart rate, hyperactive use of muscles, and a cold skin [72]. In scientific literature it has been recognised as well that a patient monitoring system can help increase the life quality of the patient and decrease the risks that are inherent to the disease [72]. Multiple sensors and wearable technologies exist that can collect patient data relevant for the patient monitoring process [72]:

- Electrocardiogram (ECG) sensors monitor the heart.
- Respiratory sensors check the breathing rate.
- Skin temperature sensors monitor the skin temperature.
- Muscular Electromyography (EMG) sensors monitor the muscle activity.

All these sensors collect measurements on the patient's health. Note that a single sensor or even a few sensors combined are not enough to capture the COPD. For instance, the patient might take a walk outside in the winter and a sensor registers a low skin temperature. In that case, the patient is not necessarily suffering from COPD at that moment. However, an expert can build patient-specific decision rules to capture COPD in such a monitoring system. For instance, if the sensors register a low skin temperature, a short and fast breathing rate, together with high blood pressure, the monitoring process might decide that the patient is suffering an attack and running out of oxygen.

A suitable way of modelling such decisions in complex environments is through the Decision Model and Notation (DMN) standard, since the standard provides maintainability and understandability of decisions [77, 78, 106]. Figure 7.2 gives an example decision requirements graph of a DMN model for COPD severeness based on data gathered by the sensors. Figure 7.3 provides the top-level COPD decision table.

7.5 Decision Model Change Patterns

To accommodate changes, designers should be able to evolve the decision models. A number of changes can occur in the decision model. The core elements of a decision model are depicted in

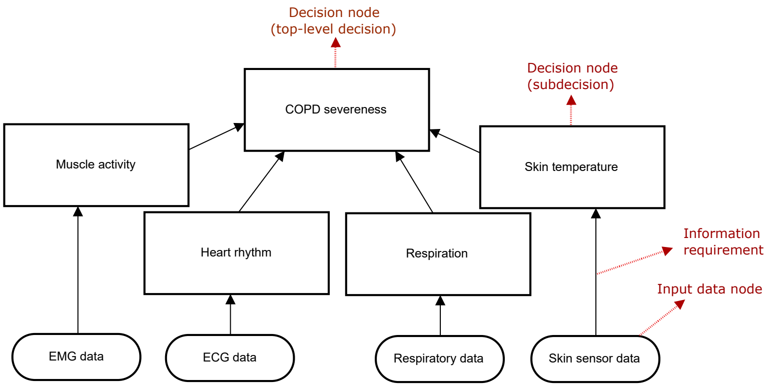
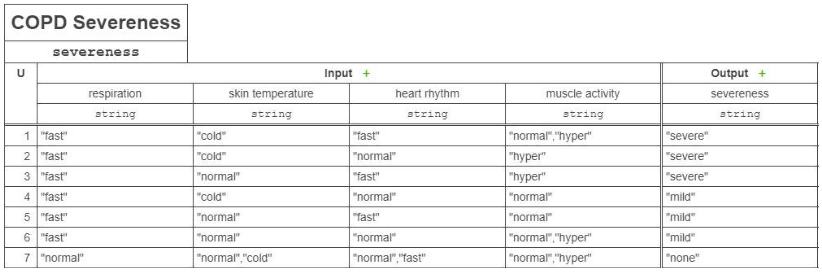


Figure 7.2: A DRD model for COPD severeness based on data from the sensors.



COPD Severeness

severeness

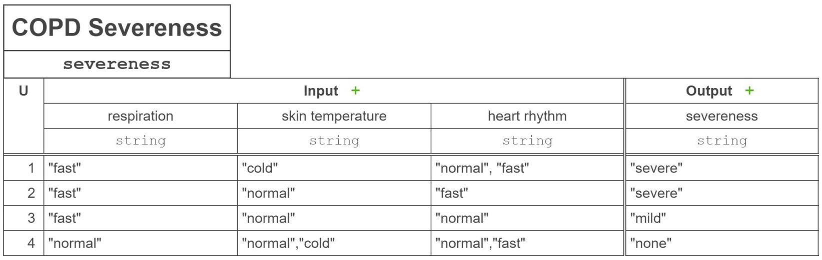
U	Input				Output
	respiration	skin temperature	heart rhythm	muscle activity	severeness
	string	string	string	string	string
1	"fast"	"cold"	"fast"	"normal","hyper"	"severe"
2	"fast"	"cold"	"normal"	"hyper"	"severe"
3	"fast"	"normal"	"fast"	"hyper"	"severe"
4	"fast"	"cold"	"normal"	"normal"	"mild"
5	"fast"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

Figure 7.3: A decision table for COPD severeness.

Figure 7.2, 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 7.3. To determine the change patterns, we investigate the changes that can manifest themselves on core DMN elements, provided in the metamodel of the DMN specification [106], at different levels of granularity. An overview of the full DMN metamodel is provided in Figure C.1 in Appendix C, followed by more detailed meta models for the distinctive parts of DMN. First, we assess the change patterns within a single decision rule, i.e., changing the inputs and outcomes of a single rule. 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 metamodel (see Figure C.2). Furthermore, we investigate change patterns on the topological structure of the DRD itself, i.e., the addition and deletion of core DRD elements (decision nodes and data input nodes), as specified in the metamodels in Figures C.3 and C.4. The change patterns are derived from the formalisation of core decision model elements in Section 7.3 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 [148]. We illustrate each change pattern on the running example introduced in Section 7.4. Table 7.1 provides an overview of the change patterns directly relating to core DMN elements.

7.5.1 Decision table change patterns

For changing decision rules, we can distinguish a plethora of change patterns. We denote a change pattern with $\Delta\Pi$, Π refers to a pattern, while Δ stands for change, e.g., ΔI_d denotes a change in the input variables of a decision node. In essence, three elements in the decision table can undergo changes, as derived from Definition 7.3: the inputs I_d, the outputs O_d and the logic L mapping the inputs to the outputs, i.e. the decision rules. We define these changes in what follows.


COPD Severeness

severeness

U	Input +			Output +
	respiration	skin temperature	heart rhythm	severeness
	string	string	string	string
1	"fast"	"cold"	"normal", "fast"	"severe"
2	"fast"	"normal"	"fast"	"severe"
3	"fast"	"normal"	"normal"	"mild"
4	"normal"	"normal","cold"	"normal","fast"	"none"

Figure 7.4: Decision table with changed inputs.

7.5.1.1 Changes within decision rules

Changing inputs

ΔΠ 1 (Decision input exclusion)**.** *A change ΔI_{d_-} indicates a change in the input set I_d of a decision D as follows: an input variable, or an existing value of a variable, can be deleted from a decision table.*

***ε* 1** (Example)**.** Suppose that in the top-level decision table represented in Figure 7.3, the input of *muscle activity* is considered irrelevant for the decision on COPD severeness. Hence, this input variable is entirely deleted from the decision table. Figure 7.4 presents the decision table after change pattern ΔΠ1 was applied. Note that this decision table has been refactored to avoid overlapping and missing rules, since simply deleting an input variable can render the decision table to be incomplete and incorrect [31].

ΔΠ 2 (Decision input inclusion)**.** *A change ΔI_{d_+} indicates a change in the input set I_d of a decision D as follows: opposite to* ΔΠ*1, an input variable, or a new value for an existing variable, can be added to a decision table.*

***ε* 2** (Example)**.** Adding a new input variable results in the exact opposite changes as in the previous change pattern: if the input variable of *muscle activity* were to be added again to the decision table in Figure 7.4, that would result in the decision table in Figure 7.3.

Changing outcomes

ΔΠ 3 (Decision output inclusion)**.** *A change ΔO_{d_+} indicates a change in the output set O_d of a decision D as follows: a new output value can be added to a decision table.*

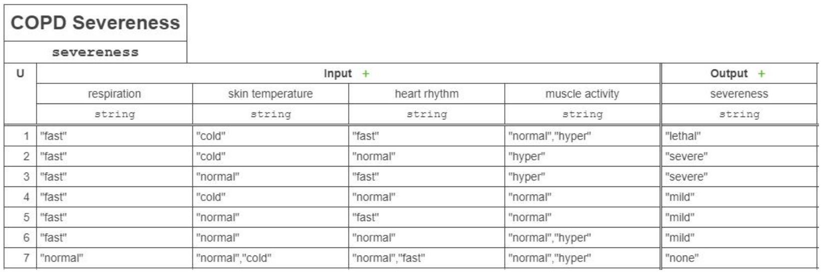
***ε* 3** (Example)**.** Suppose that in the table presented in Figure 7.3, the COPD severeness decision outcome of the first decision rule changes from *severe* to *lethal*. This new table is given in Figure 7.5. Notice that the outcome value *lethal* is a new outcome that was not represented in the decision table before.

ΔΠ 4 (Decision output exclusion)**.** *A change ΔO_{d_-} indicates a change in the output set O_d of a decision D as follows: an existing output value can be deleted from a decision table.*

***ε* 4** (Example)**.** Suppose that in the table presented in Figure 7.5, the COPD severeness decision outcome of the first decision rule changes from *lethal* to *severe*, i.e., the outcome of *lethal* is deleted from the decision table. The table in Figure 7.5 is consequently reverted to the situation in Figure 7.3.

Changing decision logic

ΔΠ 5 (Decision rule logic change)**.** A change ΔL indicates a change in the logic L of a decision D, i.e., a change in relating the existing input symbols I_d to the existing output symbols O_d within the decision table.



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"	"lethal"
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"

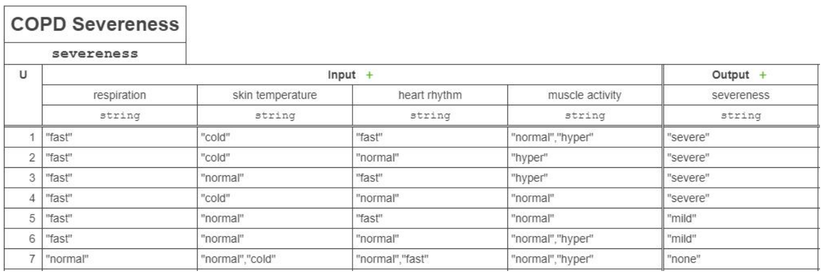
Figure 7.5: Decision table with changed outcomes.

***ε* 5** (Example)**.** Suppose that in the table presented in Figure 7.3, the logic in the fourth decision rule changes. Instead of mapping the input values to an output of *mild*, the inputs are now considered to be of a *severe* nature. This change is exemplified in the decision table in Figure 7.6.

7.5.1.2 Changes on decision rules in their entirety

ΔΠ 6 (Decision rule exclusion)**.** *If a decision rule $i_d \in I_d \xrightarrow{L} o_d \in O_d$ is deemed irrelevant at a certain point in time, it can be deleted in its entirety from a decision table.*

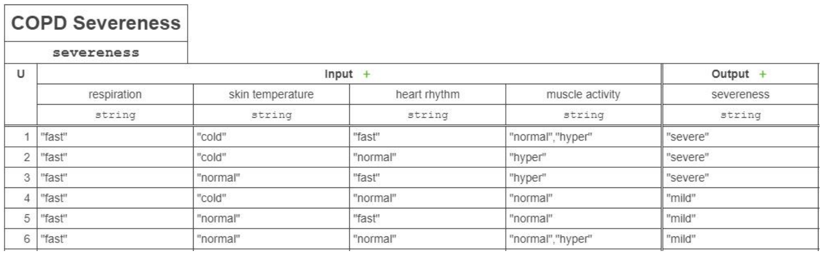
***ε* 6** (Example)**.** Suppose that decision rule 7 is entirely deleted from the decision table in Figure 7.3, rendering the decision table depicted in Figure 7.7. Notice that the decision table does not anymore contain the output *none*, and that by deleting the decision rule, the decision table is not complete anymore, i.e., the combination of input values that was present in rule 7 in Figure 7.3 can perhaps still manifest itself in real life. However, the decision table is not able to return an outcome for this input combination. This could lead to a deadlock in the system. To avoid this, either the decision table should be completed and the input values at hand should be mapped to other existing decision outcomes, or the system should be redesigned to capture the possibility of no decision outcome being returned.


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"	"severe"
5	"fast"	"normal"	"fast"	"normal"	"mild"
6	"fast"	"normal"	"normal"	"normal","hyper"	"mild"
7	"normal"	"normal","cold"	"normal","fast"	"normal","hyper"	"none"

Figure 7.6: Decision table with changed logic.



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"

Figure 7.7: Decision table with a deleted decision rule.

ΔΠ 7 (Decision rule inclusion)**.** *If a new decision rule $i_d \in I_d \xrightarrow{L} o_d \in O_d$ is deemed relevant at a certain point in time, it can be added in its entirety to an existing decision table.*

***ε* 7** (Example)**.** Suppose that the decision rule deleted in the previous change pattern is reintroduced again to the decision table. Thus, to the decision table in Figure 7.7, one rule is added, rendering the decision table in Figure 7.3.

7.5.2 DRD change patterns

This subsection deals with the deletion or addition of entire elements in the DRD model. Here, we focus on the core elements of a DRD model, i.e., decision nodes and input data nodes. By deleting the input data and decision nodes from the DRD, the information requirement arrows that connect them are deleted as well.

7.5.2.1 Decision node changes

ΔΠ 8 (Decision node exclusion)**.** *A decision node $D \in D_{dm}$ can be deleted from the DRD. This corresponds to deleting all decision rules ($I_d \xrightarrow{L} O_d$) from a decision node D. Hence, this change pattern is an aggregation of multiple* ΔΠ*6 changes. Note that deleting a decision node D also deletes all its incoming and outgoing information requirements arrows from the set IR.*

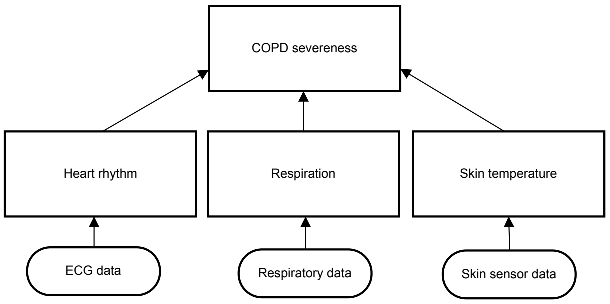


Figure 7.8: DRD adapted to changed inputs in the top-level decision.

ε 8 (Example)**.** Suppose that the subdecision `Muscle activity` in the DRD in Figure 7.2 is excluded from the model. The updated decision model is then given in Figure 7.8.

ΔΠ 9 (Decision node inclusion)**.** *A new decision node D can be added to the set of decision nodes D_{dm}. This corresponds to adding a new decision table, and thus, adding multiple decision rules ($I_d \xrightarrow{L} O_d$) encapsulated in the decision node D. Hence, this change pattern is in essence an aggregation of multiple $\Delta\Pi 7$ changes. Note that adding a decision node D also adds the necessary incoming and outgoing information requirements arrows to the set IR.*

ε 9 (Example)**.** Suppose that a decision node `Muscle activity` is added to the decision requirements diagram in Figure 7.8, effectively producing the DRD presented in Figure 7.2, as the *EMG data* input is added to the DRD as well.

7.5.2.2 Input data node changes

ΔΠ 10 (Input data node inclusion)**.** *A new data input node can be added to the set of data input nodes ID. By adding a data input node, its necessary input requirement arrows are also added to the set of IR and connected to the relevant decision nodes in D_{dm}. 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.*

ε **10** (Example)**.** Consider the decision model provided in Figure 7.8. Assume that an additional input data node is added that provides information on muscle activity. The new decision model is then given in Figure 7.9.

$\mathbf{\Delta\Pi}$ **11** (Input data node exclusion)**.** *A data input node can be deleted from the set of data input nodes* ID*. By deleting a data input node, all its input requirement arrows are also deleted from to the set of* IR*. 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.*

ε **11** (Example)**.** Consider the decision model provided in Figure 7.9. Assume that the `Muscle activity` input data node is deleted from the requirements diagram since this information is now deemed invalid to make the top-level `COPD severeness` decision. The newly obtained DRD is then given in Figure 7.8.

7.5.3 Change patterns on non-core DMN elements

So far, we have enumerated change patterns that have a considerable impact on the core DMN elements according to the metamodel presented in the standard specification [106] (see

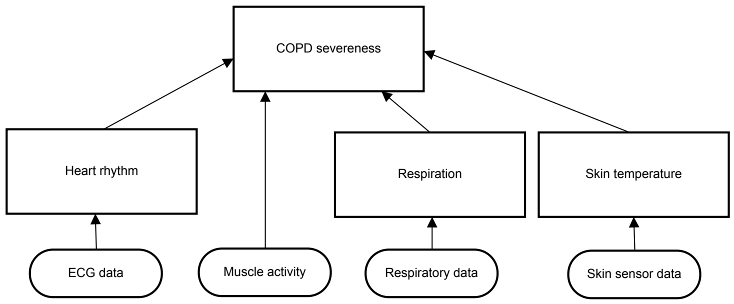


Figure 7.9: A DRD with an added input data node.

Table 7.1: 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.

Appendix C). These are: the input data nodes, the decision nodes, and the underlying decision tables and their components. However, changes related to non-core elements of the model are possible as well. We give an overview of those changes. First, we enumerate the changes that are in essence the same as the core changes presented in this section, since the elements that undergo the changes are either composed out of the elements that have already been discussed, or they connect the previously discussed core elements.

– Business knowledge models (BKM) and associated knowledge requirements can be introduced in a DRD and, therefore, they can undergo changes as well. The encapsulated decision logic of business knowledge models is often represented in the form of decision tables as well (see the metamodel in Figure C.5). Hence, the change patterns the BKMs can go through are analogous to those of the decision nodes and their related decision tables.

– Decision services are built up of core elements of the DRD, as shown in the metamodel in Figure C.6. Hence, changing the decision services corresponds to changing the individual core elements within the DRD that are encapsulated in the decision services. As such, these changes have been addressed above. Note that according to the latest DMN standard specification [106], there is an element which can be used in the DRD to represent a decision service. This decision service element encapsulates a subset of the core elements of a DRD, much like a subprocess task in BPMN encapsulates a subset of logically related process elements of a master process.

– The requirement connections between DRD elements can change, i.e., the requirement arrows can be added or deleted. This has possible impact on both the begin and end nodes associated to the requirement arrows, as shown in the decision metamodel in Figure C.3. If the requirement arrow is deleted and the begin node has no other requirements associated to it, it needs to be deleted as well. If a new requirement arrow is added, the relevant information passed through that arrow should be incorporated in the designated end node.

Next to these non-core DMN elements that are closely related to the core DMN elements, other DMN components exist as well, as specified in the metamodels in Appendix C, and, therefore, they too can be subject to change.

– The data types of the input and/or outputs can be changed (see Figure C.2).

– The names of all the elements within the decision model can be changed (see Figure C.1).

– Knowledge sources and associated authority requirements can be added to the DRD, as shown in the metamodels in Figures C.3 and C.5, and they can undergo change as well. However, the knowledge source and authority requirements in DMN are mainly used for documentation purposes and do not affect the rest of the model.

– Text annotations and associations can be added, deleted, or changed to provide additional textual information on elements within the DRD, as depicted in the metamodel in Figure C.7. These are meant for documentation and clarification purposes and do not affect the decision model in any way if they are subject to change.

– The hit policies of decision tables can be altered as well, as shown in the decision table metamodel in Figure C.2. This usually leads to refactoring and remodelling the decision table [31], however, it has no further influence on the decision model elements from the input side or the output side. When the hit policy of the decision table is changed, as the decision rules will often need to undergo reordering to fit the newly assigned hit policy. However, researchers have argued to stick with unique hit decision tables and to avoid hit policies such as first hit and priority hit in order to avoid ambiguity and overlapping rules as a design principle [49, 147].

– Note that refactoring and simplifying decision tables does, however, change how they represent logic. Modelling logic with decision tables has been discussed in previous works, in particular when it comes finding overlapping and missing rules in decision tables, and on refactoring the decision tables to remedy these issues [8, 91]. Furthermore, decision table simplification was researched as well [31, 40]. Here, the goal is to represent the decision logic in a decision table in an unambiguous and understandable way by for instance contracting the inputs of decision rules which lead to the same decision outcome. As such, the notions of decision table refactoring and simplification are well researched and they have led to tools that support the automatic refactoring and simplification of decision tables.

7.6 Change Propagation

Applying a single change pattern to a decision model can have repercussions across multiple elements of a DRD. More precisely, an applied change pattern can render the decision model inconsistent. To restore consistency other change patterns may need to be triggered. In what follows, we investigate how each change pattern should trigger a chain of change patterns within the decision model. For each change pattern, we state which inconsistencies the change may induce into the decision model. Next, we indicate which change patterns need to be triggered and propagated to resolve the inconsistencies. We provide an overview of change pattern propagation in Table 7.2.

7.6.1 ΔΠ1-induced change pattern propagation

7.6.1.1 Induced inconsistencies

Deleting an input from a decision rule (ΔΠ1) corresponds to deleting the input requirements of that decision table. The input of a decision consists of its input data elements and/or the outcomes of its subdecisions. Hence, deleting the input of the decision table without deleting its predecessors that supply the inputs, renders the decision model inconsistent.

7.6.1.2 Resolving inconsistencies

Given the hierarchical topology of a DRD, ΔΠ1 implies changing the elements that are required by the decision whose inputs undergo changes:

- If the deleted input is obtained from a data input node, this manifests itself by deleting input data nodes (ΔΠ11).
- If the deleted input was obtained from another decision node, i.e., a subdecision, the propagated change patterns to the subdecisions can be manifold.
  - An existing decision outcome in the subdecision table can be deleted (ΔΠ4). Note that this only applies if the subdecision outcome is not used as an input in another decision table. Otherwise, other decision tables that require the subdecision outcome are rendered inconsistent.
  - Another option is to exclude a whole decision rule in the subdecision whose decision outcome feeds into the decision table with the changed input (ΔΠ6).

  – If an entire input variable is deleted, rather than just an existing value of an existing input variable, the entire subdecision node that feeds its outcome into the higher-level decision where the input is deleted in the first place (ΔΠ8).

Thus, applying ΔΠ1 can on its turn trigger change patterns ΔΠ4, ΔΠ6, ΔΠ8, and ΔΠ11. Take for instance the deletion of the variable *muscle activity* from the inputs in the decision table in Figure 7.3, rendering decision table 7.4. This also affects the decision hierarchy in the DRD in Figure 7.2. The subdecision `Muscle activity`, together with its input data, i.e., *EMG data*, are not part of the top-level decision anymore. As such, they should be discarded from the decision requirements diagram. In this case, ΔΠ1 also triggers change patterns ΔΠ8 and ΔΠ11. The updated decision requirements model is provided in Figure 7.8.

7.6.2 ΔΠ2-induced change pattern propagation

7.6.2.1 Induced inconsistencies

Adding an input to a decision rule (ΔΠ2) corresponds to changing the input requirements of that decision table. The input of a decision consists of its input data elements and/or the outcomes of its subdecisions. Hence, adding an input to the decision table without changing its predecessors that supply the inputs, renders the decision model inconsistent.

7.6.2.2 Resolving inconsistencies

Given the hierarchical topology of a DRD, ΔΠ2 implies changing the elements that are required by the decision whose inputs undergo changes:

– If the added input is obtained from a data input node, this manifests itself by adding input data nodes (ΔΠ10).

– If the added input is obtained from another decision node, i.e., a subdecision, the propagated change patterns to the subdecisions can be manifold.

- A new decision outcome in the subdecision table can be added ($\Delta\Pi3$).
- Another option is to include a whole new decision rule in the subdecision whose decision outcome feeds into the decision table with the changed input ($\Delta\Pi7$).
- If a new input variable is introduced, rather than just a new value for existing input variables, a whole new decision node that feeds its outcome into the higher-level decision where the input is added can be included($\Delta\Pi9$).

Thus, applying $\Delta\Pi2$ can on its turn trigger change patterns $\Delta\Pi3$, $\Delta\Pi7$, $\Delta\Pi9$, and $\Delta\Pi10$. Take for instance decision table 7.4 and suppose that the *muscle activity* variable is added again. The DRD should reintroduce the subdecision `Muscle activity` and its required *EMG data.* Thus, this change pattern version would return the models to the initial state as shown in Figures 7.2 and 7.3, by initiating change patterns $\Delta\Pi9$ and $\Delta\Pi10$ after $\Delta\Pi2$.

7.6.3 $\Delta\Pi3$-induced change pattern propagation

7.6.3.1 Induced inconsistencies

Given the hierarchical topology of a DRD, adding a decision outcome to a decision rule ($\Delta\Pi2$) also affects the input requirements, i.e. the inputs, of the higher level decision table that requires the added decision outcome, assuming that the added decision outcome does not belong to the top-level decision node of the DRD. Simply changing the outcome of a decision without adapting the higher-level decision tables that require that outcome, leads to an inconsistent decision model.

7.6.3.2 Resolving inconsistencies

Applying $\Delta\Pi3$ can trigger change pattern $\Delta\Pi2$ in all higher-level decision tables that require the decisions affected by $\Delta\Pi3$. Note that newly added outcome values can be captured in higher-level decision tables by introducing new decision rules ($\Delta\Pi7$), or even new decision nodes ($\Delta\Pi9$) if an entirely new decision outcome variable is introduced. Take for instance the decision table represented in Figure 7.3, the COPD severeness decision outcome of the first decision rule changes from *severe* to *lethal*. This new table is given in Figure 7.5. Notice that in this case the DRD in Figure 7.2 does not undergo any change, since the changed decision outcome belongs to the top-level decision of the DRD. Hence, the outcome does not affect any decision constructs that are higher in the decision hierarchy.

7.6.4 ΔΠ4-induced change pattern propagation

7.6.4.1 Induced inconsistencies

Given the hierarchical topology of a DRD, deleting a decision outcome to a decision rule (ΔΠ4) also affects the input requirements, i.e. the inputs, of the higher level decision table that requires the deleted decision outcome, assuming that the deleted decision outcome does not belong to the top-level decision node of the DRD. Here too, simply changing the outcome of a decision without adapting the higher-level decision tables that require that outcome, leads to an inconsistent decision model.

7.6.4.2 Resolving inconsistencies

Applying ΔΠ4 should trigger change pattern ΔΠ1 in all higher-level decision tables that require the decisions affected by ΔΠ4. Similar to the previous change pattern, deleted outcome values can be resolved by deleting decision rules that require the outcome values in higher-level decision tables ((ΔΠ6), or deleting entire higher-level decision nodes if the outcome variable is deleted in its entirety (ΔΠ8) It is worthwhile to notice the exact opposite changes in the model occur as in the previous change pattern. If in the decision table of Figure 7.5, the *lethal* outcome were again to be replaced by the *severe* outcome, then the decision table would undergo redesign to revert back to the model presented in the running example, i.e., Figure 7.3.

7.6.5 ΔΠ5-induced change pattern propagation

Since ΔΠ5 is only concerned with the logic within one decision node of the decision model, no additional change patterns are propagated as a result of applying $\Delta\Pi5$. This is due to the fact that the input and outcome sets of the decision table do not change, but merely the logic that maps the inputs onto the outcomes. The inputs and outcomes are linking elements for the information requirements that construct the decision hierarchy. Since this change pattern does not alter the inputs and outcomes, no change propagation to other elements in the decision model can occur as long as the set of input values and the set of output values remain the same.

7.6.6 $\Delta\Pi6$-induced change pattern propagation

7.6.6.1 Induced inconsistencies

Excluding a decision rule ($\Delta\Pi6$) is in essence a combination of deleting inputs ($\Delta\Pi1$), deleting decision outcomes ($\Delta\Pi4$), and the logic that connects them ($\Delta\Pi5$). Hence, this change pattern can possibly induce the same model inconsistencies as patterns $\Delta\Pi1$ and $\Delta\Pi4$.

7.6.6.2 Resolving inconsistencies

Thus, applying $\Delta\Pi6$ can on its turn trigger change patterns $\Delta\Pi1$, $\Delta\Pi4$, $\Delta\Pi5$ and $\Delta\Pi6$ again. Notice that if no input variables or rule outcomes are deleted when $\Delta\Pi6$ occurs, then this pattern corresponds to $\Delta\Pi3$, i.e., a mere change in underlying logic, without any further change propagation.

When applying $\Delta\Pi6$ to decision rule 7 in the decision table in Figure 7.3, obtaining the decision table depicted in Figure 7.7, a unique decision output value, i.e., *none*, is deleted. However, this does not affect the rest of the decision model, since the affected decision table corresponds to the top-level decision. As such, these change patterns cannot propagate any patterns affecting the higher levels in the decision hierarchy. Furthermore, the lower-level elements in the decision hierarchy are not affected either, since the deleted input values are not unique, and no input variable was deleted in its entirety. Thus, in this case, $\Delta\Pi6$ only triggers a change in underlying logic ($\Delta\Pi5$) without affecting the hierarchy of the DRD. However, the decision table in Figure 7.7 is not longer complete and should undergo redesign according to [31] to capture all possible input combinations.

7.6.7 $\Delta\Pi7$-induced change pattern propagation

7.6.7.1 Induced inconsistencies

Including a decision rule ($\Delta\Pi7$) is, in analogy with $\Delta\Pi6$ propagation, a combination of $\Delta\Pi2$, $\Delta\Pi3$, $\Delta\Pi5$ in the case of deleting existing values of existing input variables or outcomes. Hence, this change pattern can possibly trigger the same inconsistencies as patterns $\Delta\Pi2$ and $\Delta\Pi3$ when inputs or outcomes are added to a decision rule, assuming new values of existing variables are introduced.

7.6.7.2 Resolving inconsistencies

Thus, applying $\Delta\Pi7$ can on its turn trigger change patterns $\Delta\Pi2$, $\Delta\Pi3$, $\Delta\Pi5$ and $\Delta\Pi7$ again. Notice that if no unique values of input variables and rule outcomes are introduced when $\Delta\Pi7$ occurs, then this pattern corresponds to $\Delta\Pi5$, i.e., a mere change in underlying logic, without any further change propagation.

When applying $\Delta\Pi7$ by decision rule 7 in the decision table in Figure 7.7, obtaining the decision table depicted in Figure 7.3, a unique decision output, i.e., *none*, is added. As such the table is complete. However, in analogy with $\Delta\Pi6$, no other elements in the decision hierarchy undergo any changes in this example.

7.6.8 $\Delta\Pi8$-induced change pattern propagation

7.6.8.1 Induced inconsistencies

Deleting an entire decision node ($\Delta\Pi8$) corresponds to applying $\Delta\Pi6$ on every decision rule contained within the decision table of that decision node, i.e. deleting the decision table related to that node. As such, the same chain of inconsistencies is established. All elements that are lower in the decision hierarchy and that are only connected to the deleted decision node, become obsolete as well. The elements that are higher in the decision hierarchy and that require the outcome of the deleted decision node, render the model inconsistent, since the required inputs can no longer be obtained due to the deletion of the lower-level decision node.

7.6.8.2 Resolving inconsistencies

All input nodes, i.e., input data nodes and subdecisions, that feed into the deleted decision node and are only connected to that decision node, need to be deleted, effectively triggering $\Delta\Pi 8$ and $\Delta\Pi 11$. This applies to decision model elements that are lower in the decision hierarchy than the deleted decision node. For elements that are higher in the decision hierarchy, change pattern $\Delta\Pi 1$ should be triggered as well, i.e., whole input variables are deleted, since the required decision node that provides those variables is deleted as well. As such, the underlying decision logic of the decision model can be subject to changes as well $\Delta\Pi 5$.

Suppose that the subdecision `Muscle activity` in the DRD in Figure 7.2 is excluded from the model. This also deletes the subdecision's input data, i.e., *EMG data* ($\Delta\Pi 11$). The updated decision model is then given in Figure 7.8. Since the deleted decision node is connected to a higher-level decision node, i.e., `COPD severeness`, which requires the deleted decision node as input, inconsistencies arise in the decision model. Namely, the decision table in Figure 7.3 has a *muscle activity* input that can no longer be obtained due to the deletion of the `Muscle activity` decision node. As such, $\Delta\Pi 8$ also triggers $\Delta\Pi 1$ in this example, effectively transforming the decision table in Figure 7.3 into the decision table given in Figure 7.4 by deleting the input variable *muscle activity*. As such, the decision model is rendered consistent again.

7.6.9 $\Delta\Pi 9$-induced change pattern propagation

7.6.9.1 Induced inconsistencies

Inserting an entire decision node ($\Delta\Pi 9$) corresponds to adding an entirely new decision table, i.e., applying $\Delta\Pi 7$ on every decision rule contained within the decision table of that decision node. Hence, the same inconsistencies arise as in $\Delta\Pi 7$.

7.6.9.2 Resolving inconsistencies

For all decision elements that require the inserted decision node, a new variable will need to be introduced in the inputs ($\Delta\Pi 2$). For the inserted decision node itself, the relevant input requirements need to be provided, either by linking existing decision nodes and input data nodes to the inserted decision node, or by introducing new decision nodes and input data nodes (ΔΠ9 and ΔΠ10). This also affects the decision logic (ΔΠ5).

Suppose that the subdecision `Muscle activity` is added to the DRD in Figure 7.8, obtaining the DRD in Figure 7.2. This also adds the subdecision's input data, i.e., *EMG data* (ΔΠ10) to the DRD model in Figure 7.2. Furthermore, the decision table in Figure 7.4 corresponding to the `COPD severeness` decision needs to be altered as well to incorporate the addition of the new decision node. As such the *muscle activity* variable is added to reflect the information requirement between the `Muscle activity` subdecision and the `COPD severeness` decision (ΔΠ2). This way, the decision table in Figure 7.3 is obtained. As such, the consistency between the different decision model elements is restored.

7.6.10 ΔΠ10-induced change pattern propagation

7.6.10.1 Induced inconsistencies

Including a new input data node (ΔΠ10) in a DRD renders the decision model inconsistent if the decision table to which the input data node is connected, is not updated as well to consider that input node.

7.6.10.2 Resolving inconsistencies

Including a new input data node (ΔΠ10) also corresponds to changing the inputs of all decision rules in all decision nodes that require the added input data node (ΔΠ2). This also changes the decision logic (ΔΠ5). Hence, all decision nodes that require the input data node need to introduce the variable concerning the newly added input data node to their inputs.

Consider the decision model provided in Figure 7.8 with its corresponding top-level decision table in Figure 7.4. Assume that an additional input data node is added that provides information on muscle activity. The new DRD model is then given in Figure 7.9. However the newly introduced *Muscle activity* node induces a change in the decision table in Figure 7.4 as well (ΔΠ2). More precisely, the inputs of that decision table need to change to incorporate the *muscle activity* variable, rendering the decision table in Figure 7.3. As such, the consistency between the decision model elements is restored.

7.6.11 $\Delta\Pi11$-induced change pattern propagation

7.6.11.1 Induced inconsistencies

Excluding an input data node ($\Delta\Pi11$) from a DRD renders the decision model inconsistent if the decision table to which the input data node was connected, is not updated as well.

7.6.11.2 Resolving inconsistencies

Excluding an input data node ($\Delta\Pi11$) corresponds to changing the inputs of all decision rules in all decision nodes that require the deleted input data node ($\Delta\Pi1$). This also changes the decision logic ($\Delta\Pi5$). Hence, the variable derived from the deleted input data node is deleted from all the decision nodes that required the input data node. Since the input data nodes are by definition (see Definition 7.1) the lowest elements in the decision hierarchy, they can only propagate their changes to higher-level decision nodes. This holds both for change propagation induced by $\Delta\Pi11$, as well as $\Delta\Pi10$.

Consider the decision model provided in Figure 7.9 with its corresponding top-level decision table in Figure 7.3. Assume that the `Muscle activity` input data node is deleted. The new DRD model is then given in Figure 7.8. However, the deleted *Muscle activity* node should induce a change in the decision table in Figure 7.3 as well ($\Delta\Pi1$). More precisely, the inputs of that decision table need to change to incorporate the deletion of the *muscle activity* variable, rendering the decision table in Figure 7.4. As such, the consistency between the decision model elements is restored.

7.6.12 Overview of induced change pattern propagation

Notice that all the applied change patterns can also induce a change in the decision logic ($\Delta\Pi5$), i.e., the mapping between input and output elements. When a change pattern is applied to the decision model, it can trigger another change pattern. That triggered change pattern can in turn trigger yet another change pattern. Thus, a chain of triggered patterns may need to be propagated through the decision model until the system stabilises. Table 7.2 gives an overview of how this change pattern propagation chain can come into existence. Figure 7.10 provides a depiction of the change propagation graph. For instance, when pattern ΔΠ1 is applied, ΔΠ8 may need to be triggered. Change pattern ΔΠ8 can in turn trigger ΔΠ11 after which the propagation of changes stop and the system returns to a stable state. Note that, as indicated in Table 7.2, the change patterns can also induce a change in decision logic (ΔΠ5). However, we have excluded ΔΠ5 from the change propagation graph in order to render the graph more comprehensible.

Table 7.2: Change propagation: an overview of decision model change patterns that may need to be triggered in other elements of the decision model to restore consistency.

Applied ΔΠ	**Propagated ΔΠ possibly triggered by initial ΔΠ**
ΔΠ*1*	ΔΠ*4*, ΔΠ*5*, ΔΠ*6*, ΔΠ*8*, ΔΠ*11*
ΔΠ*2*	ΔΠ*3*, ΔΠ*5*, ΔΠ*7*, ΔΠ*9*, ΔΠ*10*
ΔΠ*3*	ΔΠ*2*, ΔΠ*5*, ΔΠ*7*, ΔΠ*9*
ΔΠ*4*	ΔΠ*1*, ΔΠ*5*, ΔΠ*6*, ΔΠ*8*
ΔΠ*5*	
ΔΠ*6*	ΔΠ*1*, ΔΠ*4*, ΔΠ*5*, ΔΠ*6*
ΔΠ*7*	ΔΠ*2*, ΔΠ*3*, ΔΠ*5*, ΔΠ*7*
ΔΠ*8*	ΔΠ*1*, ΔΠ*5*, ΔΠ*8*, ΔΠ*11*
ΔΠ*9*	ΔΠ*2*, ΔΠ*5*, ΔΠ*9*, ΔΠ*10*
ΔΠ*10*	ΔΠ*2*, ΔΠ*5*
ΔΠ*11*	ΔΠ*1*, ΔΠ*5*

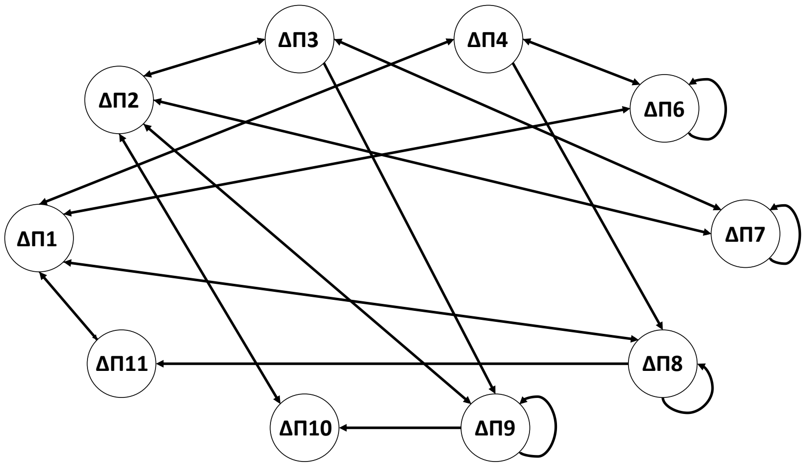


Figure 7.10: Overview of change propagation: change patterns that can possibly follow each other. Note that ΔΠ5 should be considered as a consequence of applying any change pattern, but it is not included in the graph in order to provide a more comprehensible overview.

7.7 A Modelling Environment for DMN Model Evolution

This section presents a modelling environment prototype which provides modelling tool support for the evolution of DMN decision models. The modelling environment is based on the open source Camunda[13] modeller which we have advanced with verification capabilities. The modeller can apply any of the change patterns discussed in this chapter. After doing so, the modelling environment checks for consistency errors and displays error messages. The modelling environment can highlight the errors and suggests actions to remedy the inconsistencies. After the modeller selects an action, the modelling environment automatically performs it and checks for errors again. This way, the DMN can be evolved iteratively in a consistent way. In what follows, we explain the interface of the modelling environment and we show its verification capabilities with an example.

[13]`https://camunda.com/download/modeler/`

7.7.1 The modelling environment

Figures 7.11 and 7.12 show the interface of the DMN modelling environment, with numbers indicating the different parts of the tool, which are as follows:

1. Display of the DRD model.

2. Buttons to upload a DMN model into the environment, to download the model from the environment, to generate an empty DMN model, and to connect the modelling environment to an existing model repository.

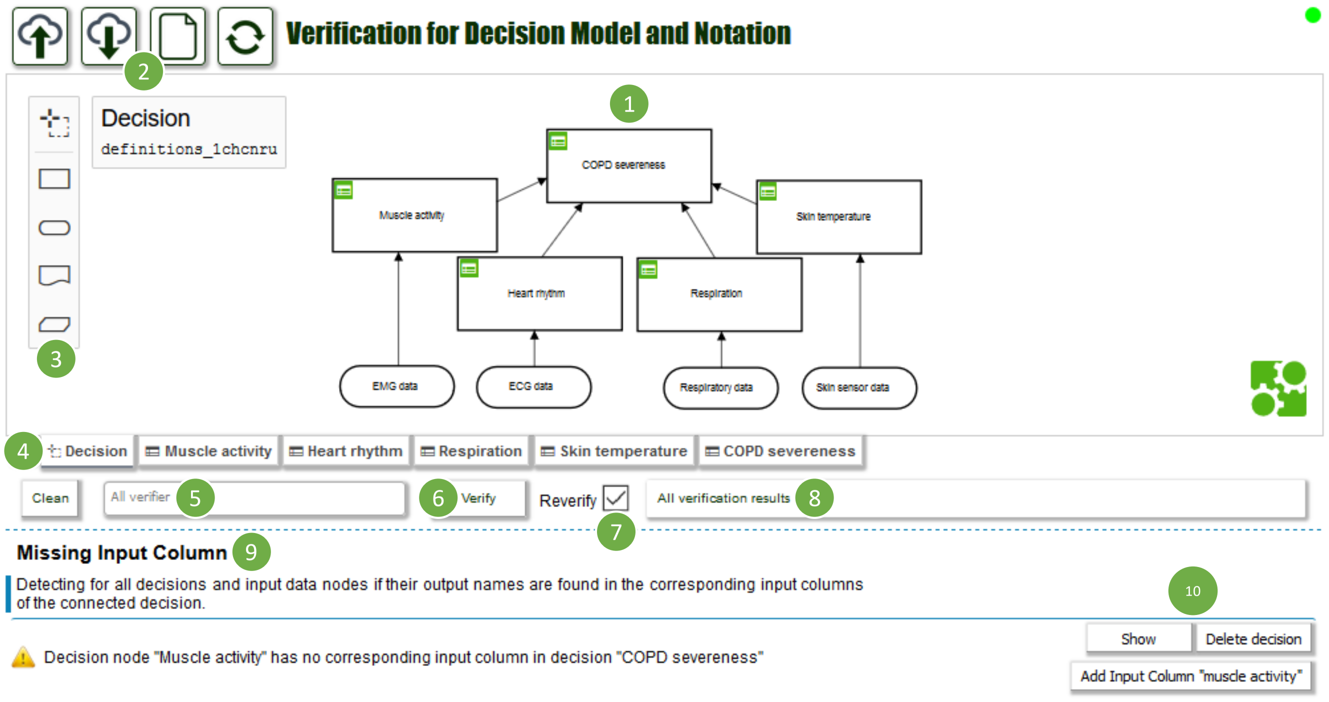
3. DMN elements which can be selected by the modeller to add to the DMN model.

4. The modeller can switch between the different views of the DMN model, e.g., by selecting a decision, the tool switches from the DRD view in Figure 7.11 to the decision table view in Figure 7.12.

5. Different verfication mechanisms are offered. The modeller can select the verifiers which are relevant, i.e., the errors that the modelling environment should check for. Alternatively, instead of selecting a single verifier, all verifiers can be selected as to display all errors that the modelling environment discovers.

6. The *Verify* button. By clicking this button the modelling environment checks for errors (specified in the previous point) in the current DMN model.

7. The *Reverify* check box. By checking this box the modelling tool will perform a verification of the model every time the modeller performs one of the change patterns on the model. Unchecking this box deactivates this feature, as an experienced modeller may not need to check the consistency after every action performed. However, after applying a whole chain of changes, the modeller can again re-enable the verification feature of the tool.

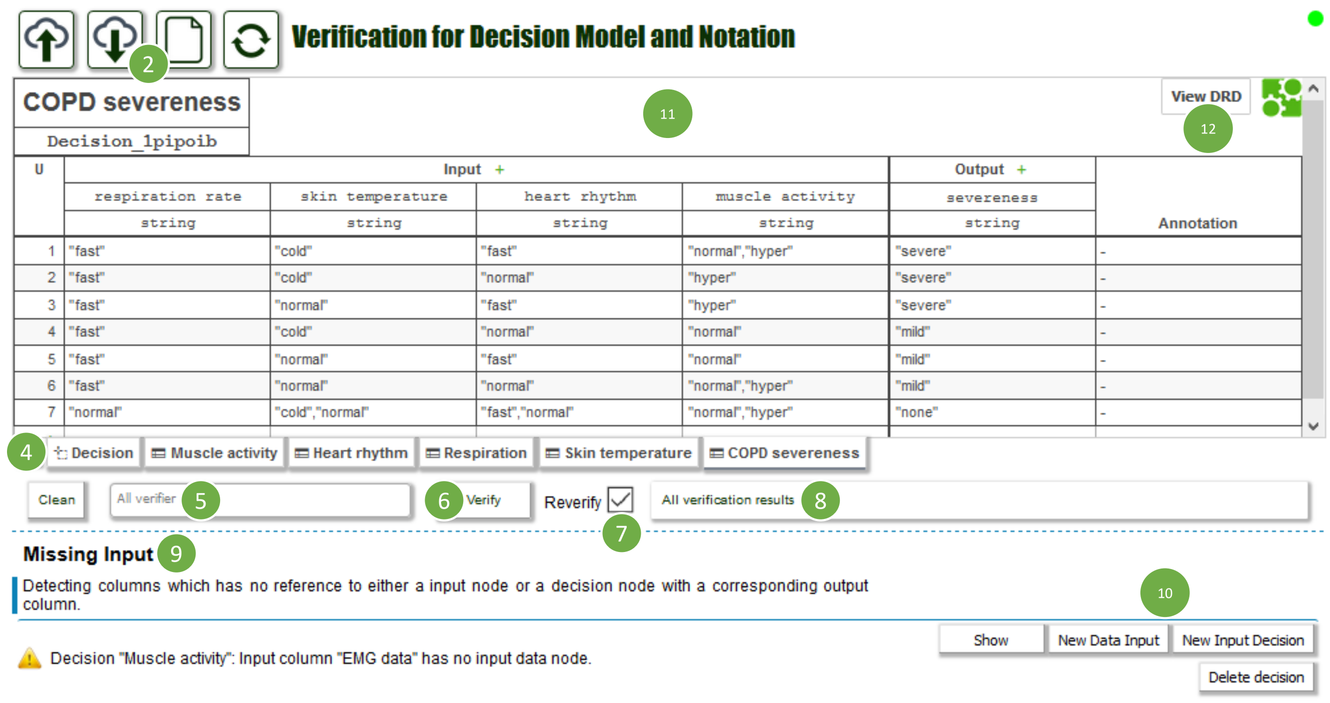


Missing Input Column

Detecting for all decisions and input data nodes if their output names are found in the corresponding input columns of the connected decision.

Decision node "Muscle activity" has no corresponding input column in decision "COPD severeness"

Figure 7.11: The DRD view of the DMN verification tool.



COPD severeness

Decision_1pipoib

U	Input				Output	Annotation
	respiration rate	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"	"cold","normal"	"fast","normal"	"normal","hyper"	"none"	-

Missing Input

Detecting columns which has no reference to either a input node or a decision node with a corresponding output column.

⚠ Decision "Muscle activity": Input column "EMG data" has no input data node.

Figure 7.12: The decision table view of the DMN verification tool.

8. Drop-down menu to display the verification results. After the verification happens, the modeller can select which errors, i.e., which verification results to display. Alternatively, the modeller can opt to display all verification results.

9. Feedback for the modeller that explains the errors. Here, the error messages as a result of the verification mechanism of the modelling environment are displayed. The error messages are explained as well, both in general terms, as well as in specific terms, i.e., indicating the names of the DMN elements that are affected.

10. Action buttons to highlight or resole the errors. Next to the specific error messages, automatic actions are suggested by the modelling environment that can resolve the errors. By clicking the *Show* button, the modelling environment highlights the errors in the DMN model in a red colour. By clicking any of the other suggested action buttons, the modelling environment automatically performs the selected action.

11. Display for the decision table view. Instead of the DRD model, a decision table that has been selected is displayed in the central view of the tool, as shown in Figure 7.12.

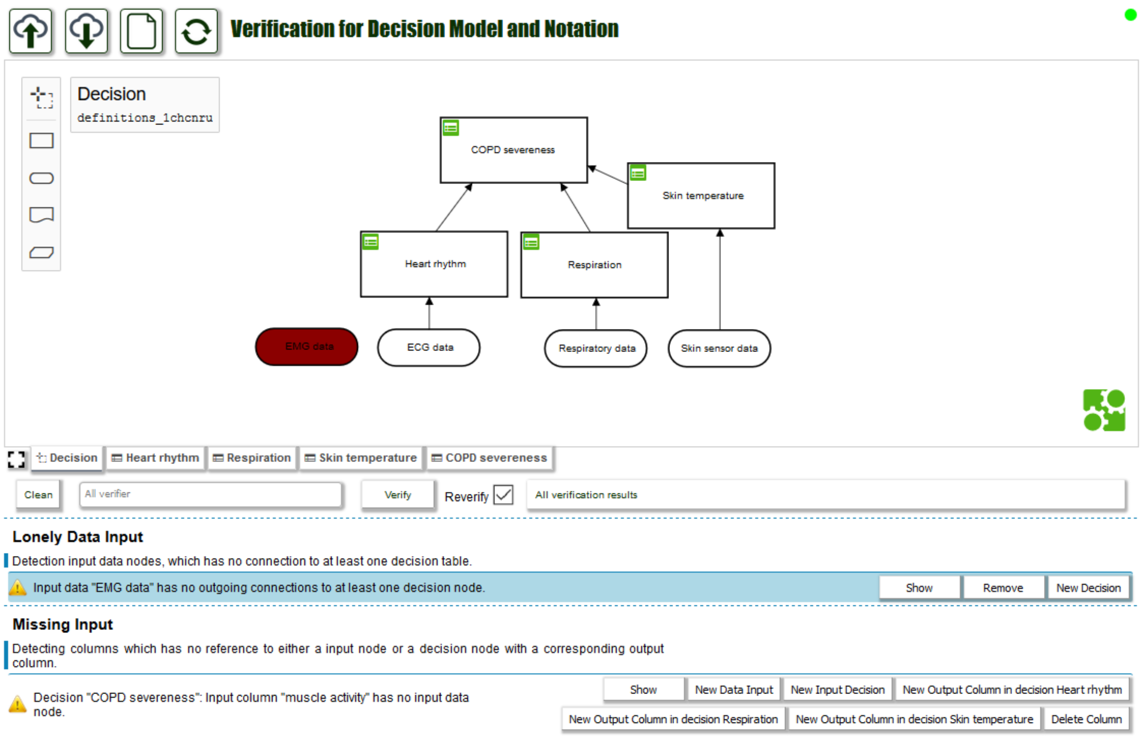
12. The *View DRD* button. By clicking this button the modeller can switch back from the decision table view to the DRD view in the modelling environment.

7.7.2 Example of change propagation as supported by the modelling environment

In this subsection we provide a short example to show how the DMN verification tool works. The DMN model presented in Figures 7.2 and 7.3 is uploaded into the modelling environment. Suppose that instead of assessing muscle activity through the *Muscle Activity* subdecision with data obtained from the EMG sensor, the physician wishes to investigate the patient and to manually enter the muscle activity classification. Hence, the *Muscle Activity* subdecision from Figure 7.2 is obsolete and the modeller decides to delete it (i.e., the modeller applies decision node exclusion). The modelling environment raises two errors: *Lonely Data Input*, i.e., an input data node which is not connected to any decision node, and *Missing Input*, i.e., an input column in the top-level *COPD Severeness* decision which does not have any input data node or subdecision node providing the required input. Figure 7.13 shows the situation after applying this change pattern and clicking the *Show* button for the unconnected input data node.

By clicking the *Remove* button next to the input data node error, the change pattern of input data node exclusion is automatically performed. The modelling environment then re-verifies the model to check for new inconsistencies (arising through the application of the change pattern), leaving only the missing input error relating to the top-level *COPD Severeness* decision. Figure 7.14 highlights this error in the DRD view of the modelling environment, while Figure 7.15 highlights the error in the decision table view.

As shown in Figures 7.14 and 7.15, the modelling environment suggests a number of actions to remedy the issues surrounding the missing input in the top-level decision. Inputs can be secured in three different ways: adding a new data input (i.e., input data inclusion), adding a new input decision (i.e., decision node inclusion), or adding outputs to existing subdecision nodes (i.e. decision output inclusion). Alternatively, the situation can be remedied by deleting the input column which does not have the required inputs (i.e., decision input exclusion). After deciding that a physician will manually provide the information needed as input in the *COPD Severeness* decision table, the action of adding new data input is selected and the modelling environment automatically introduces a matching input data node, yielding a consistent model as show in Figure 7.16.



Lonely Data Input

Detection input data nodes, which has no connection to at least one decision table.

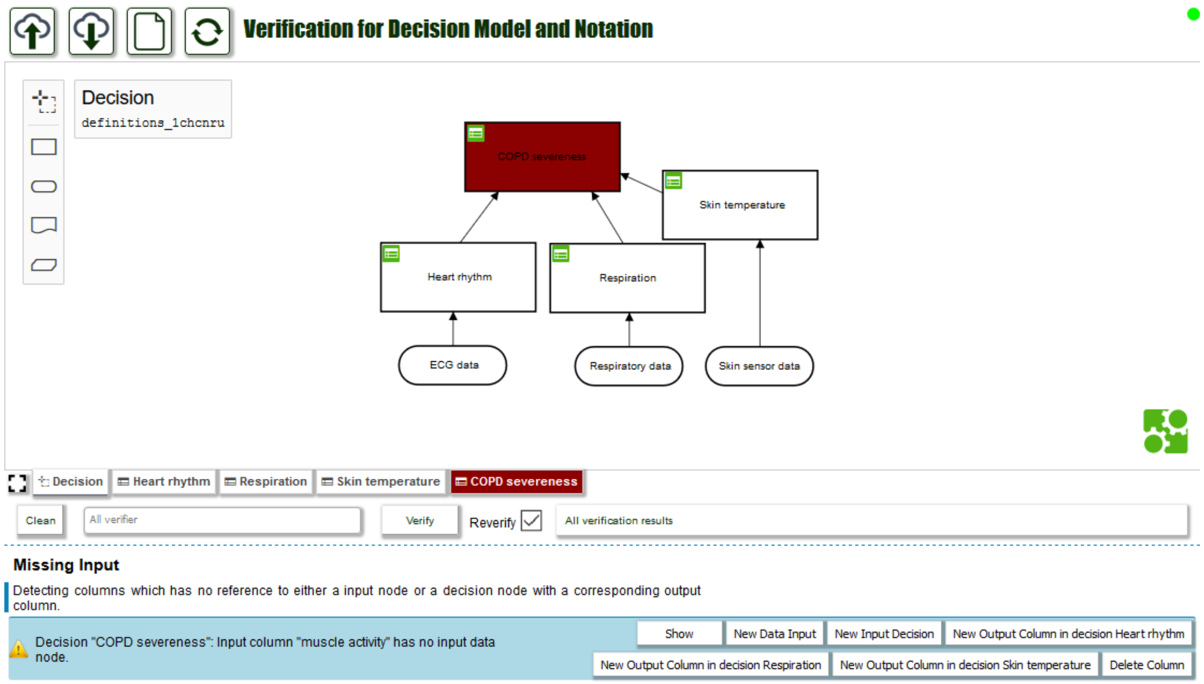
Input data "EMG data" has no outgoing connections to at least one decision node.

Missing Input

Detecting columns which has no reference to either a input node or a decision node with a corresponding output column.

Decision "COPD severeness": Input column "muscle activity" has no input data node.

Figure 7.13: Situation after applying decision node exclusion.

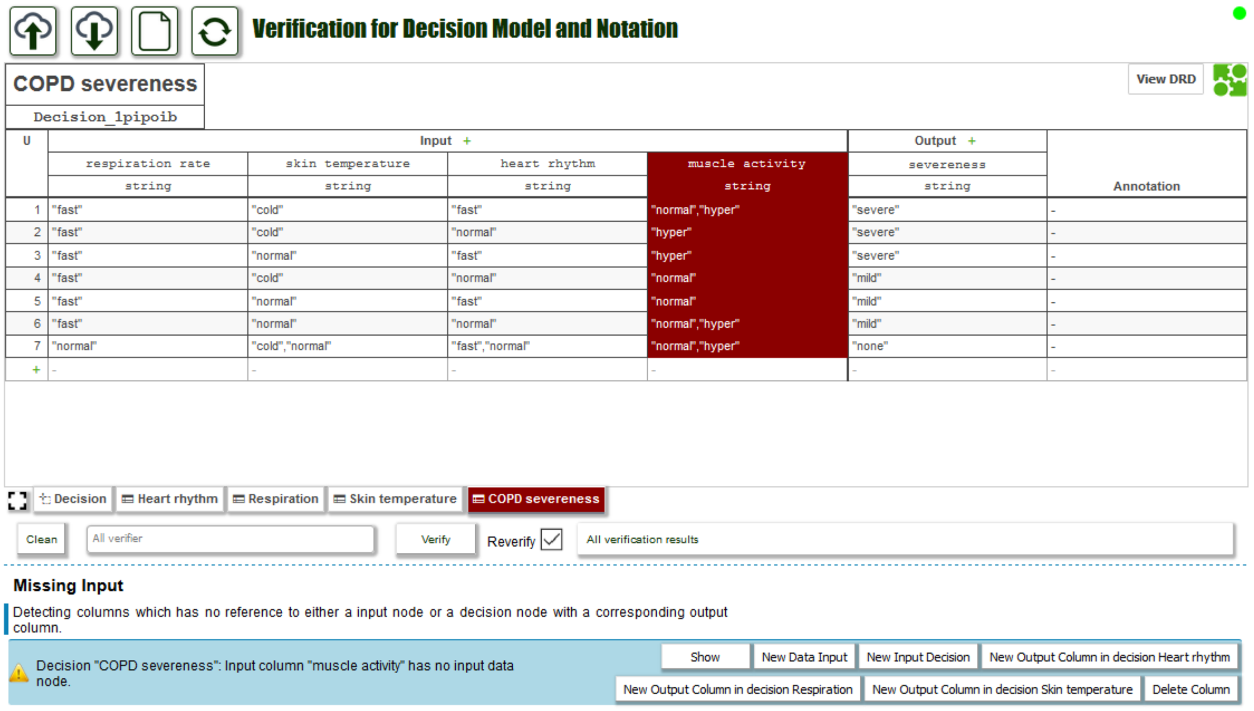


Missing Input

Detecting columns which has no reference to either a input node or a decision node with a corresponding output column.

Decision "COPD severeness": Input column "muscle activity" has no input data node.

Figure 7.14: Situation after applying input data node exclusion (DRD view).



COPD severeness

Decision_1pipoib

U	Input				Output	Annotation
	respiration rate	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"	"cold","normal"	"fast","normal"	"normal","hyper"	"none"	-
+	-	-	-	-	-	-

Missing Input

Detecting columns which has no reference to either a input node or a decision node with a corresponding output column.

Decision "COPD severeness": Input column "muscle activity" has no input data node.

Figure 7.15: Situation after applying input data node exclusion (decision table view).

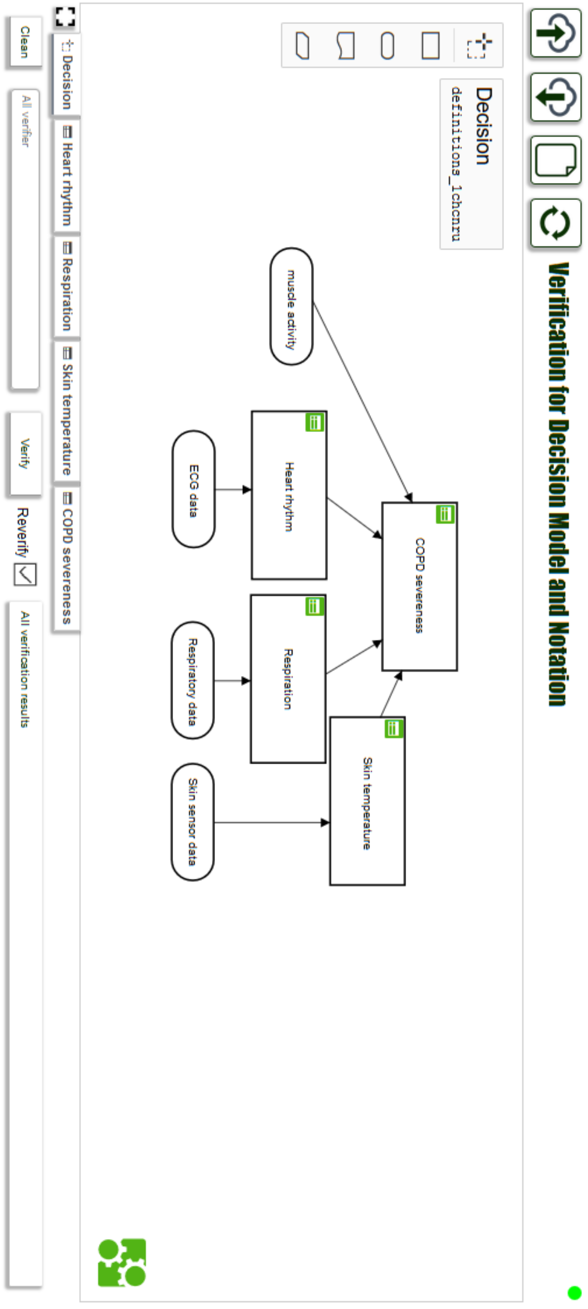


Figure 7.16: Stable situation after applying input data node inclusion.

7.8 Conclusion and Future Work

This chapter investigates and defines possible decision model change patterns for the evolution of DMN decision models. Through a running example we have illustrated that the adaptation of DMN decision model can lead to within-model inconsistencies. In order to remedy these inconsistencies, additional changes may need to be propagated throughout the entire decision model, effectively triggering other change patterns. This way, consistency between the different decision model elements is upheld. The proposed change patterns illustrate how decision models can evolve and which inconsistency concerns rise up when evolving the DMN models. Furthermore, the change patterns can aid in developing and implementing flexible and dynamic decision management systems in order to move away from the static approaches present both in literature and in industry. We illustrate the feasibility of the approach by providing a proof-of-concept modelling environment prototype which provides modelling support and automation for change pattern administration on DMN decision models.

In future work, we will investigate how changing decision models impact 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 (BPMN) and decision (DMN) model environment. Changing the underlying decisions of a process can lead to a construction of a new process variant. Additionally, in the case of knowledge-intensive processes, instead of BPMN, change patterns can be considered for Case Management Model and Notation (CMMN) models, as they allow to model activities that can be performed in an unpredictable order by knowledge workers. Furthermore, it should be investigated whether DMN can be extended with a decision ontology to better cope with knowledge-intensive aspects that cannot adequately be represented in a decision table [30]. Moreover, flexible decision models are of particular interest to Internet-of-Things (IoT) process settings [56, 83], as IoT process are inherently subject to a dynamic and changeable environment.

CHAPTER 8

Performance Assessment of the Modelling Environment for DMN Model Evolution

"Nothing has such power to broaden the mind as the ability to investigate systematically and truly all that comes under your observation in life."

Meditations, Book III
— *Marcus Aurelius*

This chapter was published as follows:

Faruk Hasić, Carl Corea, Jonas Blatt, Patrick Delfmann, Estefanía Serral Asensio. A Tool for the Verification of Decision Model and Notation (DMN) Models. `14th International Conference on Research Challenges in Information Science (RCIS), Demo Session`, Limassol (Cyprus), 2020.

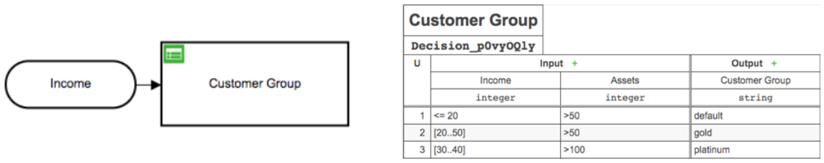
Abstract. The Decision Model and Notation (DMN) is a decision modelling standard consisting of two levels: the decision requirement diagram (DRD) level which depicts the dependencies between elements involved in the decision model, and the decision logic level, which specifies the underlying decision logic, usually in the form of decision tables. As the decision tables and DRD are modelled in conjunction, the need to verify the consistency of both levels arises. While there have been some works geared towards the verification of decision tables, the DRD-level has been strongly neglected. In this work, we therefore present a tool for the model verification of DMN models at both the logic and the DRD level, along with the performance assessment of the tool.

8.1 Introduction

The Decision Model and Notation has at its the core the business rule decision logic usually modelled by means of decision tables (*decision logic level*). The standard can also provide decision requirement diagrams (*DRD level*), which allow specifying the requirements and relations of decisions involved in the decision model [42, 62, 66].

While DMN standardizes how to represent decision logic and decision requirements, no means are provided to guide experts in consistently modelling the two levels. In result, modelling errors can easily occur. This can be seen in Figure 8.1, which show an inconsistent DMN decision model. First, there are errors within the decision logic, specifically, the income conditions for rules 1 and 2 are overlapping (when the input is exactly 20). Second, the DRD level and the decision table are inconsistent because the *Assets* requirement is missing on the DRD level.

Such modelling errors are a problem currently faced in practice, as proven by interviews with key practitioners who unanimously report problems in verifying the correctness of DMN models [124]. Also, empirical results on detected modelling errors within the decision logic of a large insurance company are presented in [11]. This calls for automated means to support companies in the verification of decision models.



Customer Group

Decision_p0vyOQly

U	Input +		Output +
	Income	Assets	Customer Group
	integer	integer	string
1	<= 20	>50	default
2	[20..50]	>50	gold
3	[30..40]	>100	platinum

Figure 8.1: DMN model: DRD (left) and decision table (right) with modelling errors.

Table 8.1 shows an overview of existing DMN verification approaches. As can be seen from the DRD level verification capabilities in Table 8.1, recent research is sparse. In a previous work [40], we presented means to detect a selection of decision logic level verification capabilities as proposed in the business rule management capability framework [124]. In the work at hand, we present novel verification capabilities for the DRD level, as well as some novel decision logic level capabilities derived from [69]. To the best of our knowledge, our tool is the first to provide such an extensive set of verification mechanisms. As can be seen from Table 8.1, this work extends the verification capabilities of [40] with the following verification capabilities: *unused predefined value verification, missing predefined value verification, missing in-/output value verification, missing in-/output column verification, idle data input data verification, missing input verification, multiple input verification* and *inconsistent type verification.* We will discuss these capabilities in detail in Section 8.2. We refer to the work in [40] for the decision logic level verification capabilities that were part of that work and are also incorporated in the tool presented here.

8.2 Tool Description and Usage Example

The developed tool relies on the camunda-dmn[14] library and is available for demonstration[15]. The tool allows to upload and analyse DMN models directly in the browser. The tool presented in this work extends the one presented in [40] with the following novel decision logic and DRD level verification capabilities derived from [69]:

8.2.1 Decision logic capabilities (verification of decision logic within a decision table):

– **Missing predefined value verification.** Detecting if there is a value in a rule which is not part of the predefined values.

– **Unused predefined value verification.** Detecting whether there are predefined values for a column, but no rule for that value.

– **Missing input value.** Detecting whether there is an output value of another table (where that table is an input to the

[14]`https://github.com/camunda/camunda-engine-dmn`

[15]`https://bit.ly/2OEPEpH`

Table 8.1: Overview of verification capabilities covered by existing approaches (X = full and o = partial support).

		Decision logic level											**DRD level**					
Literature		Identical Rules	Equivalent Rules	Subsumed Rules	Indeterminism	Overlapping Conditions	Partial Reduction	Missing Rules	Unused Predefined Value	Missing Predefined Value	Missing input value	Missing output value	Missing input column	Missing output column	Idle data input	Missing (data) input	Multiple (data) input	Inconsistent types
Calvanese et al. (2016)	[29]	X	o	X		X		X										
Laurson et al. (2016)	[91]	X		X	o	X	o	X										
Batoulis et al. (2017)	[8]	X		X		X		X										
Calvanese et al. (2017)	[30]	X	o	X	o	X		X										
Ochoa et al. (2017)	[104]										X	X						
Batoulis et al. (2018)	[7]	X		X		X		o										
Calvanese et al. (2018)	[31]	X	o	X	o	X	o	X										
Corea et al. (2019)*	[40]	X	X	X	X	X	X	X										
This work		**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**

table under consideration), but the table does not have a corresponding input value (missing rule).

- **Missing output value.** Detecting whether there is an input value in a table, but the subdecision does not have a corresponding output value (and thus, the current rule would be unreachable).

8.2.2 DRD level capabilities (verification of interconnection between logic modules, i.e., between different decision tables in the DRD, as well as between input data and decision tables):

- **Missing input column verification.** Given a decision and a subdecision, detecting whether the subdecision has an output column, but the decision table of the higher-level decision does not have a corresponding input column.
- **Missing output column verification.** Given a decision and a subdecision, detecting whether the higher-level decision has an input column, but the subdecision table does not have a corresponding output column.
- **Idle data input verification.** There is a data input node that is not used.
- **Missing (data) input verification.** There is (data or subdecision) input that is used in a decision, but it is missing in the DRD.
- **Multiple (data) input verification.** For any input column, detecting whether there are too many inputs.
- **Inconsistent type verification.** Given a decision and a subdecision, detecting whether the corresponding columns have the same data type.

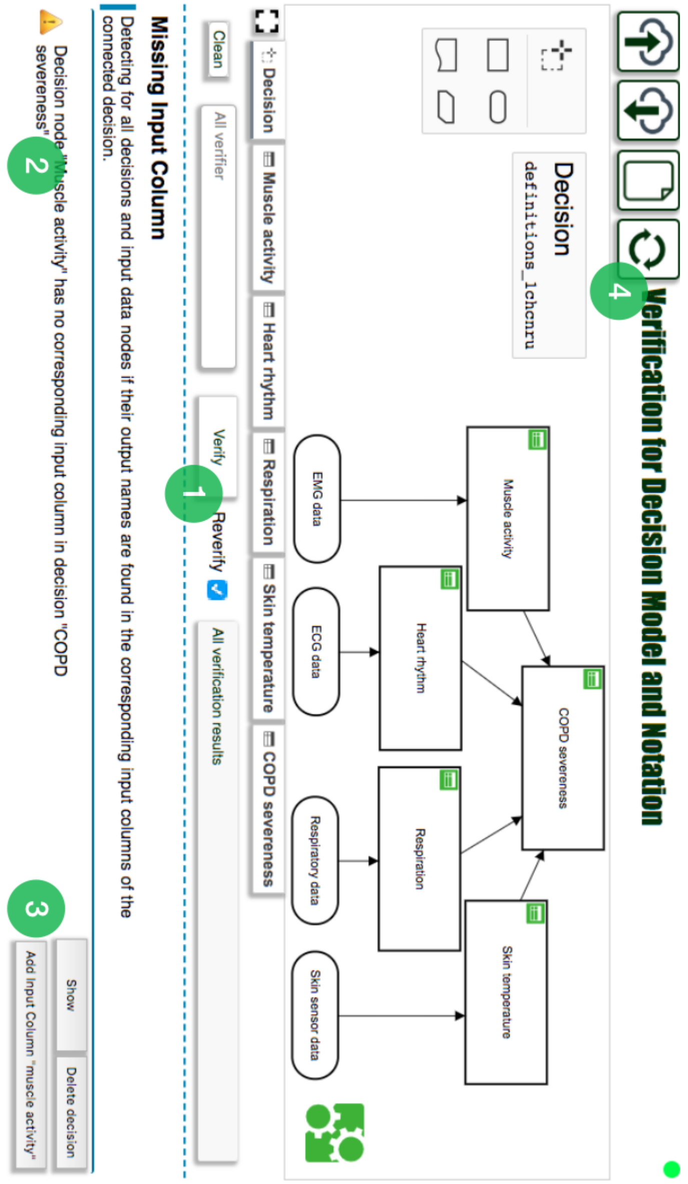
Figure 8.2 provides a usage example of the tool. The user can upload and verify a DMN model (1). Note that modellers can easily switch from the DRD-level to the table level by clicking on an individual table. The tool identifies all errors based on the shown verification capabilities and displays error messages (2). The tool can highlight the errors and allows the user to quickly browse the issues. Next to this issue, the tool suggests actions to remedy the modeling issues and allows the modeller to select one, which is then automatically performed by the tool (3). Additionally, the modeller can then re-verify the resulting model (see checkbox next to (1)). This way, the modeller can incrementally apply actions to restore a consistent model. The tool can be used to verify arbitrary DMN models, but can also be connected to workflow engines such as Camunda. Here, the user can deploy the updated and verified decision model directly from the tool (4).

8.3 Maturity and Outlook

For evaluation, we performed run-time experiments and analysed a total of 900 synthetic decision models. As parameters for generating these models, our custom generator[16] accepts the number of rows and columns in a single decision table, as well as the number of overall nodes on a DRD level. As the novel DRD level capabilities are the focus of this work, we continue discussing the results for a varying number of DRD level nodes in the model. Please see [40] for further experiments on the decision logic verification capabilities with a varying number of columns and rows.

For the experiments, we selected *{18,36,...,180}* as the number of nodes on the DRD level, and *{10,20,...,100}* as the number of rules per table from (i.e., 10x10 possible combinations). Note that most models in literature employ less than 20 nodes and at most a few tens of rules per node. For each of the 100 possible combinations of rows and nodes, we generated 9 decision models as follows: on the DRD-level, the decision tables and input nodes were connected at random, which allowed to create synthetic decision models with actual errors such as missing or redundant input data. The respective rules of these tables were generated by using random integer conditions (see [40] for details). The number of columns for each decision table ranged from 2 to 5 at random. Consequently, for

[16]An interface to use our generator can be found at `https://bit.ly/31q1U2r`



Missing Input Column

Detecting for all decisions and input data nodes if their output names are found in the corresponding input columns of the connected decision.

Decision node "Muscle activity" has no corresponding input column in decision "COPD severeness"

Figure 8.2: The DRD view of the DMN verification tool.

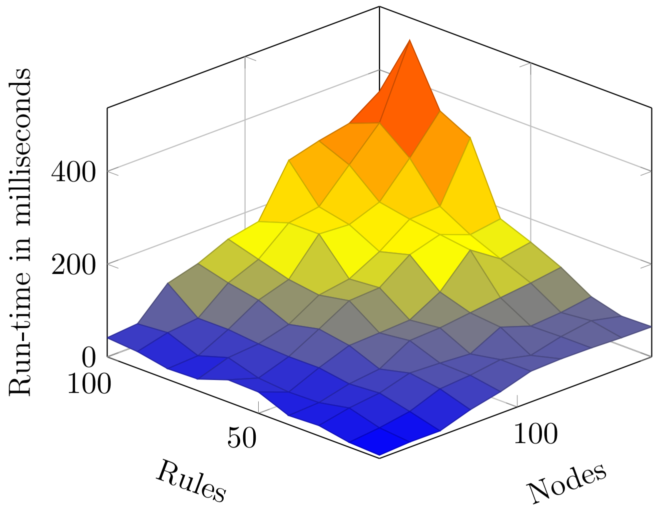


Figure 8.3: Run-time statistics for the analysed synthetic decision models with up to 180 nodes on the DRD-level and 100 rules per table.

each of the 10x10 combinations, we applied the verification tool for the 9 respective models and computed the average run-time for each parameter configuration, as shown in Figure 8.3. The experiments were run on Ubuntu Xenial with E312 processor and 16GB RAM.

As it can be seen in Figure 8.3, the run-time for analysing a model with 180 DRD nodes and 100 rules per table averages to roughly 0.4 seconds. Thus, for the analysed data set, our tool allowed for a feasible analysis.

8.4 Conclusion

The tool presented in this chapter allows to analyse DMN models both on a logic level and on a DRD level, which is a current issue faced in practice. As shown in Table 8.1, our work closes existing research gaps by implementing all depicted verification capabilities in a unified tool. Our tool supports the current DMN standard, allowing companies to perform model verification on their DMN models. Through a direct integration with workflow management systems, our tool can be used for consistent model evolution and thus facilitates sustainable business rule management. In future work, we aim to apply our tool to industrial data sets and to conduct usability studies with practitioners.

CHAPTER 9

Consistent Evolution of Process and Decision Models

"I judge pleasure and pain to be of small importance compared to knowledge, the appreciation and contemplation of beauty, and a certain intrinsic excellence of mind which, apart from its practical effects, appears to me to deserve the name of virtue. Many years it seemed to me perfectly self-evident that pleasure is the only good and pain the only evil. Now, however, the opposite seems to me self-evident."

Letter to Gilbert Murray, April 3, 1902
— *Bertrand Russel*

This chapter will be submitted for publication as follows:

Faruk Hasić, Carl Corea, Jonas Blatt, Patrick Delfmann, Estefanía Serral Asensio. Consistent Evolution of Integrated Process and Decision Models. *To be submitted for publication*, 2020.

Abstract. Business processes often rely on decision knowledge. Such decision-aware processes are often modelled through integrated Business Process Model and Notation (BPMN) and Decision Model and Notation (DMN) models. Decision knowledge is not necessarily static and can change over time. Therefore, processes should be able to cope with these changes which can occur both at the process model level as well as the decision model level. Process model change patterns have been studied extensively in literature. However, decision model change patterns are still an open field of research. This chapter discusses decision model change patterns that affect both the decision and process models in order to ensure between-model consistency. We provide a fully functioning online modelling environment that is capable of handling change patterns applied on the decision model and of automatically safeguarding the consistency between the process and decision models.

9.1 Introduction

Business processes (BPs) are chains of events, activities, and decisions, that are performed in a coordinated manner within an organisational and technical environment and ultimately add value to the organisation and its customers [44]. BPs rely on decisions to take the best line of actions at a particular context. Since complex decisions are hard to represent in a classical procedural business process model, e.g. a Business Process Model and Notation (BPMN) model [105], the Decision Model and Notation (DMN) standard [106] has recently been proposed to describe these decisions in a proper manner. Several works have already studied the integration of DMN with BPMN, often referred to as decision-aware processes [13, 24, 33, 37, 38, 66], while other have proposed tools that use the BPMN and DMN standards [25, 34, 74, 76].

Today organisations' competitiveness and continued existence depends on the ability of their processes to adapt to the different execution contexts and changing requirements. Furthermore, tools that support such changes are vital, as process change management is a challenging endeavour which cuts through different units and levels of an organisation [87]. Although several works exist that deal with BPMN model changes [4, 113, 145], and only preliminary work dealing with DMN model changes [57], no work on the support of the evolution of integrated process and decision models exists yet. Therefore, we investigate how to deal with such evolution. The contribution of this paper is twofold:

1. We analyse the influence of decision model change patterns on process and decision model consistency and determine how the consistency of the integrated models can be restored.

2. We provide proof-of-concept modelling tools that allow for applying the change patterns to evolve decision models and to automatically refactor the process and decision models to restore consistency.

This paper is structured as follows. Section 9.2 constitutes a section with a running example and related work demonstrating that current approaches do not support the problem statement as described in the running example. Section 9.3 provides preliminary formalisation. In Section 9.4 we present the change patterns based on the links between the process and decision models described in Section 9.3. Section 9.5 focuses on the propagation of decision model change patterns throughout the decision model, as well as the impact of change propagation on the process model. Section 9.6 presents the modelling environment that we have created which supports decision model change patterns and decision model evolution, together with automatic change propagation throughout the decision model as well as automatic re-integration of the process model with the newly established decision model. Finally, Section 9.7 concludes and provides directions for future research.

9.2 Running Example and Related Work

Here we constitute a running example of a decision-aware process that will be used throughout the paper. Additionally, we discuss current approaches in the literature dealing with process evolution, as well as the lack of research on the evolution of decision-aware processes.

9.2.1 Running example of a decision-aware process

Consider a patient health monitoring system for a person diagnosed with the Chronic Obstructive Pulmonary Disease (COPD). COPD is a disease that obstructs the lungs and obstructs the airflow and breathing of the patient. Acute attacks of the disease can happen. In that case the patient can experience uncomfortable complications such as fast breathing, a fast heart rate, hyperactive use of muscles, and a cold skin [72]. In scientific literature it has been recognised as well that a patient monitoring process based on Internet of Things (IoT) can help increase the life quality of the patient and decrease the risks that are inherent to the disease [72]. Multiple sensors and wearable technologies exist that can collect patient data relevant for the patient monitoring process [72]:

– Electrocardiogram (ECG) sensors monitor the heart.

– Respiratory sensors check the breathing rate.

– Skin temperature sensors monitor the skin temperature.

- Muscular Electromyography (EMG) sensors monitor the muscle activity.

All these sensors collect measurements on the patient's health. IoT settings are often data- and decision-intensive. Note that a single sensor or even a few sensors combined are not enough to capture the COPD. For instance, the patient might take a walk outside in the winter and a sensor registers a low skin temperature. In that case, the patient is not necessarily suffering from COPD at that moment. However, an expert can build patient-specific decision rules to capture COPD in such a monitoring system. For instance, if the sensors register a low skin temperature, a short and fast breathing rate, together with high blood pressure, the monitoring process might decide that the patient is suffering an attack and running out of oxygen. In such a situation the process can trigger the administration of an oxygen mask to the patient. An illustrative COPD monitoring IoT process is given in Figure 9.3. First the heart rhythm of the patient is checked and the emergency alarm is activated if needed. Afterwards, the severeness of the COPD is assessed, and given the assessment, the process ends or treatments with oxygen masks or inhalers are carried out.

Note that complex decision rules are hard to represent in a classical procedural business process model, e.g. a Business Process Model and Notation (BPMN) [105] model, as BPMN has only limited capabilities of modelling decisions [79]. These kind of decisions in BPMN would be represented in a spaghetti-like process with a lot of gateways chained together. Hence, the understandability and maintainability of such a process model becomes a burden. From the spaghetti-model it is not always immediately clear which decision logic is followed, and if the decision logic or decision outcomes need to change, the maintainability becomes a burden as well in such a complicated process model. On top of that, being a procedural language, the order in which the gateways are placed greatly impacts the decision making process and the link between the sensors and the process model. Like that, the needed decision data is collected in a step wise manner and the decision making is turned into a procedural process, rather than a declarative endeavour based on decision rules.

A more suitable way of modelling such complex decisions is through the Decision Model and Notation (DMN) standard

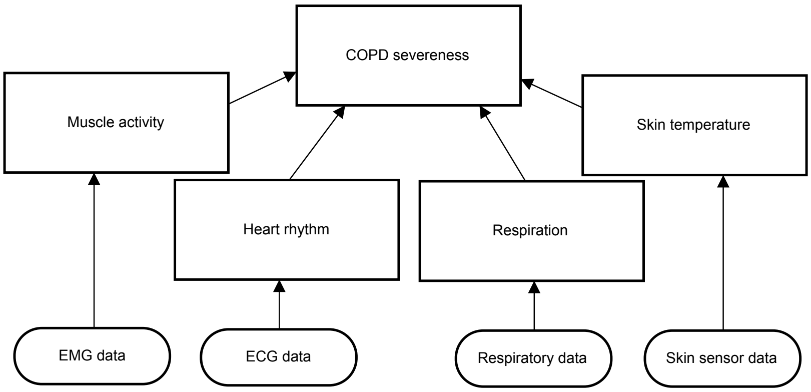
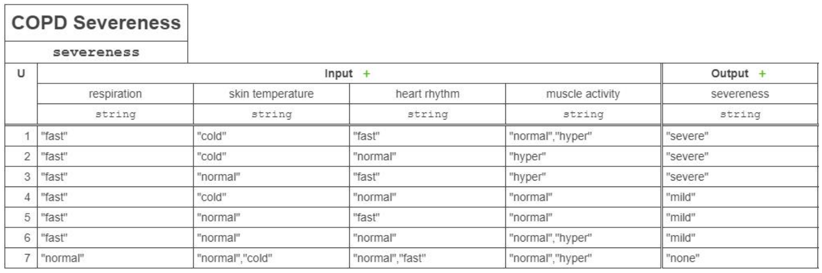


Figure 9.1: A DMN model for COPD severeness.

[106], thus securing maintainability, understandability, and the declarative nature of decisions. DMN is a recently introduced decision modelling standard that has enjoyed significant interest in literature. 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, corner-cut rectangles for business knowledge models, and ovals to represent data input. Figure 9.1 gives an example decision requirements graph of a DMN model for COPD severeness based on data gathered by the IoT sensors. Figure 9.2 provides the COPD decision table.

These decisions and decision outcomes can be integrated in the BPMN process model and consequently interpreted by the process for further execution and evolution of the process. Previous work has focused on decision-centric approaches towards process modelling. These approaches start with the construction of a DMN decision model [106] and the process model is constructed afterwards in such a way that it is automatically consistent with the provided decision model. This holistic decision approach has been described in [61, 63, 65, 66], where a strict separation of



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 9.2: A decision table for COPD severeness.

concerns between process and decision logic is adhered to. The decisions can influence the process execution in multiple ways: by affecting process conclusion and termination, by guiding the process control flow, or by influencing the data perspective of the process. Hence, decisions can be the driver of process execution as decisions can be defined in a Service-Oriented Architecture (SOA) approach which can be invoked by processes [63]. In [66], the integration of decisions and processes was addressed in more detail. Changing the underlying decisions of a process, i.e. the data perspective of the process or the context under which the process is to be executed, will also have repercussions on the process itself. Hence, by changing an underlying decision, the process that is to be executed can change as well. These changes can manifest themselves on different perspectives of the process, e.g. changes in terms of process and control flow elements, or simply changes in the data perspective of the process. Strategies towards refactoring processes to conform to a decision model were briefly discussed in [67]. Noteworthy is that the integration of BPMN and DMN was deemed useful for modelling ubiquitous business processes [149].

9.2.2 Current approaches for process evolution

With different decision models belong possibly different process executions. What kind of patient monitoring and care process will be executed depends on the decisions made based on the data obtained from the sensors. Therefore, the process must be flexible in order to adapt to this context captured by the sensors in the physical world.

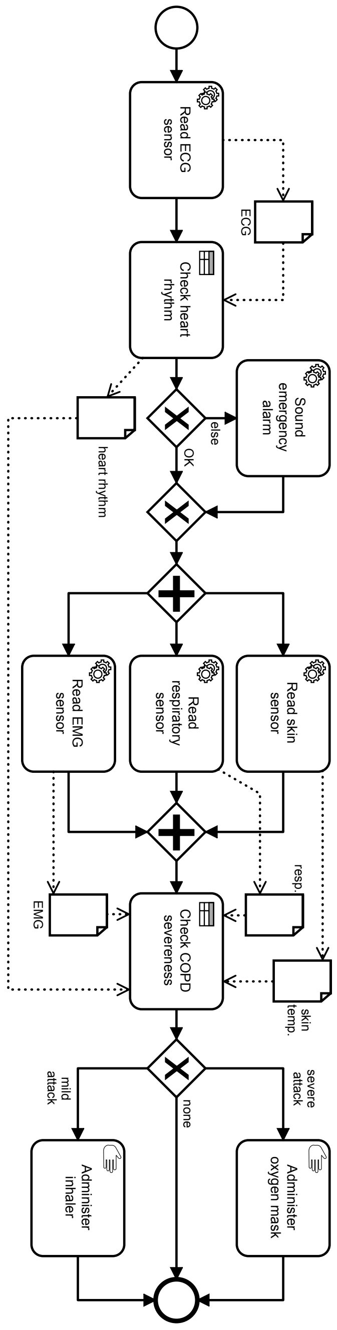


Figure 9.3: A COPD monitoring process.

The need for flexible and adapting business processes has been recognised in the process literature [2, 3, 4, 85, 111, 112, 113, 114, 119, 126, 142, 145, 149]. However, these works mainly focus on control flow change patterns, e.g., adding activities, deleting activities, changing the activity sequence, replacing activities, adding or deleting gateways, and adding or deleting control flow elements. As such, evolving the decision perspective of processes has not received sufficient attention in research, despite the inherent need of modern knowledge-based system to efficiently deal with decision rule changes [15, 26, 41, 86, 102]. Only preliminary work on DMN changes exists [57]. Therefore, we approach this research gap in this paper by introducing decision model change patterns for decision-aware process evolution, as well as by introducing a modelling environment which supports the evolution of decision-aware processes.

9.3 Preliminaries

In this section, we provide a formalisation for key DMN concepts needed for the development of the change patterns that will be discussed in the following sections. The backbone of a DMN decision model is formed by rectangles that depict decisions and subdecisions and ovals that represent data input. The underlying decision logic is usually represented in decision table form.

Definition 9.1 (Decision requirement diagram (DRD))**.** A DRD is a tuple (D_{dm}, ID, IR) consisting of a finite non-empty set of decision nodes D_{dm}, a finite non-empty set of input data nodes ID, and a finite non-empty set of directed edges IR representing the information requirements such that $IR \subseteq (D_{dm} \cup ID) \times D_{dm}$, and $(D_{dm} \cup ID, IR)$ is a directed acyclic graph (DAG).

According to the DMN standard, a decision requirement graph can be an incomplete or partial representation of the decision requirements in a decision model. The set of all DRDs in the decision model constitutes the exhaustive set of requirements. The information contained in this set can be combined into a single DRD representing the decision requirements level as a whole. The DMN standard refers to such a DRD as a decision requirement graph (DRG). We expand the notion of a DRG, in such a way that a DRG is a DRD which is self-contained, i.e. for every decision in the diagram all its requirements are also represented in the diagram.

Definition 9.2 (DRG)**.** A decision requirement diagram DRD is a decision requirement graph DRG if and only if for every decision in the diagram all its modelled requirements, present in at least one diagram, are also represented in the diagram.

The term *decision* can have a number of meanings. According to the DMN specification a decision is the logic used to determine an output from a given input. Meanwhile, in process modelling a decision is taken in a decision activity, e.g. the business rule task in BPMN. Another common meaning is that a decision is the actual result, which we call the output of a decision, or simply the decision result. We define a decision as follows:

Definition 9.3 (Decision)**.** A decision $d \in D_{dm}$ is a tuple (I_d, O_d, L), where $I \subseteq ID$ is a set of input symbols, O a set of output symbols and L the decision logic defining the relation between symbols in I_d and symbols in O_d.

In case of decision tables, I and O contain the variables of the input and output elements respectively, and L is the table itself, i.e. the set of decision rules present in the table.

In DRDs these decisions d_i are represented by the decision nodes $D_i \in D_{dm}$. We will use D to refer to both a decision and its representing node in a DRD. From the definition of DRGs, it is clear that every decision D in that model has a unique decision requirement graph DRG_D with D as its single top-level decision. A DRG contains exactly all information requirements of its top level decisions. Hence, only one DRG exists with D as its single top-level decision. We use DRG_D to denote this DRG. Furthermore, all the decisions in the DRG, except the top level decision, are consequently subdecisions of the top level decision. In other words, the top-level decision *requires* these lower level subdecisions.

Now we have formally defined the basic concepts of a DMN decision model, we need to define how these decisions manifest themselves within a business process. Decisions in processes do not surface solely as the driver of control flow. Rather, they both encompass the routing, i.e. because of decision outcomes that steer toward a certain activity tailored towards supporting its output, and the changes in the data layer of the process as well. The latter introduces numerous types of activities that are representatives of the *decision* model in the *process* model:

Definition 9.4 (Variable and activity classification)**.** The input and output data variables of activities are defined as follows:

- $I : A \rightarrow V$, function assigning activities which receive input of a certain variable,
- $O : A \rightarrow V$, function assigning activities which deliver output for a certain variable.

This enables the construction of the following activity types:

1. **Operational activities ((no) inputs, no outputs):** do not have any influence on the process' decision dimension and only act as a performer of a specific action that is tied to that specific place in the control flow. They might serve as the conclusion of a decision. They are provided with the decision inputs needed, which are not used further in the process, $A_o = \{a \in A \mid O(a) = \emptyset, \}$.
2. **Administrative activities (no inputs, outputs):** have the purpose to introduce decision inputs into the process, $A_a = \{a \in A \mid I(a) = \emptyset \wedge O(a) \neq \emptyset\}$.
3. **Decision activities (inputs, outputs):** serve a true autonomous decision purpose as they transform decision inputs into a decision outcome, $A_d = \{a \in A \mid I(a) \neq \emptyset \wedge O(a) \neq \emptyset\}$.

Note that it holds that $A_a \cup A_o \cup A_d = A$.

With the activity classification in mind, we can now make the connection with decisions in business processes and decision models. A decision in a business process can be defined as follows:

Definition 9.5 (Decision in a process model (d^a))**.** A $d^a \in D_{dm}$ is a tuple $(I_{d^a}, O_{d^a}, L_{d^a})$, where $a \subseteq A_d$, $I_{d^a} \subseteq I(a)$, $O_{d^a} \subseteq O(a)$ and $L_{d^a} \subseteq L$.

This last definition connects a decision activity with a decision in the decision model. Notice that a business process will provide the decision model with the decision it wants to invoke, and an input requirements set for that decision. This input requirements set will be provided to the decision activity $a \subseteq A_d$ which invokes the decision model in the form of $I_{d^a} \subseteq I(a)$.

9.4 Decision Model Change Patterns

To adapt to different execution contexts and requirements, designers or users should be abele to evolve the process and decision models. Process model evolution has been studied at length, as indicated in the related work (Section 9.2). However, decision model evolution is still an open subject. A number of changes can occur in the decision model. Moreover, given that the process model relies on the decision model for decisions that affect the control flow and the data perspective of the process, a change in the decision model will possibly affect the process model as well. Consistency between the process and decision models must at all times be guaranteed [66]. Hence, the process model may need to undergo adaptations in order to conform to the underlying decisions.

Notice that according to the formal definitions in the previous section, the process and decision models are loosely coupled, i.e., the process model perceives the logic in the decision model as a black box and the link between the two is established via the input and output data of the decisions. The core elements of a decision model are depicted in Figure 9.1. As can be seen from the figure, the input data and the decision nodes within a DRD are connected via information requirement arrows. The logic encapsulated in a decision node is usually modelled with decision tables, such as shown in Figure 9.2. These core elements of the DMN model are relevant to the business process. Hence we investigate the change patterns that can manifest themselves on these elements at different levels of granularity.

An overview of decision model change patterns is provided in Table 9.1. In what follows, we discuss these change patterns and their effects on process models in more detail. The first

Table 9.1: Overview of decision model change patterns from [69].

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

subsection assesses the change patterns within a single decision table, i.e., changing the preconditions and outcomes of a decision rules. Furthermore, we look at change patterns for a decision rule in its entirety, i.e., adding or deleting decision rules from a decision table. The second subsection investigates change patterns on the topological structure of the DRD itself, i.e., the addition and deletion of core DRD elements (decision nodes and data input nodes). The change patterns taken from [69] and are derived from the formalisation of decision model elements in Section 9.3 and the elementary edit operations that can be applied on the elements, i.e., insertion, deletion, and substitution, which in essence is a combination of insertion and deletion [148]. For each change pattern, we specify how the changes can impair process and decision model consistency and how the inconsistencies can be remedied. Furthermore, we illustrate the change patterns, inconsistencies, and remedies on the running example introduced in Section 9.2. Table 9.2 provides an summary of the findings in this Section.

9.4.1 Decision table change patterns

For changing decision rules, we can distinguish a plethora of change patterns. We denote a change pattern with $\Delta\Pi$, Π refers to a pattern, while Δ stand for change, e.g., ΔI_d denotes a change in the input variables of a decision node. In essence, three elements in the decision table can undergo changes, as derived from Definition 9.3: the inputs I_d, the outputs O_d and the logic L mapping the inputs to the outputs, i.e. the decision rules. We define these changes in what follows. Changes can induce inconsistencies between the process and decision models (χ), and the process itself needs to undergo changes to resolve these inconsistencies (Ψ).

9.4.1.1 Decision input exclusion

$\Delta\Pi$ 1 (Decision input exclusion)**.** *A change $\Delta I_{d\_}$ indicates a change in the input set I_d of a decision D as follows: an input variable, or an existing value of a variable, can be deleted from a decision table.*

χ 1 (Induced inconsistencies)**.** The process must provide the input data needed for decision enactment. If a variable is deleted, the variable is no longer required for decision enactment due to changes in the input set of a decision. Hence, the data objects in the process relating to the deleted variable in the decision model may become redundant data.

Ψ 1 (Resolving inconsistencies)**.** The redundant data objects relating to the deleted variables in the decision model can be discarded from the process model as well if they are not used for anything else in the process other than as input relating to the deleted decision variable.

ε 1 (Example)**.** Suppose that in the top-level decision table represented in Figure 9.2, the precondition of *muscle activity* is considered irrelevant for the decision on COPD severeness. Hence, this input variable is entirely deleted from the decision table. Figure 9.4 presents the decision table after change pattern $\Delta\Pi1$ was applied. Note that this decision table has been refactored to avoid overlapping and missing rules, since simply deleting a precondition variable can render the decision table to be incomplete and incorrect [31]. The legacy process model in Figure 9.3, still reads the muscle activity sensor and provides the data as input to the decision activity `Check COPD Severeness`. This is remedied in the redesigned process model in Figure 9.5. Furthermore, deleting a variable from the preconditions also affects the decision hierarchy in the DRD in Figure 9.1. The subdecision `Muscle activity`, together with its input data, i.e., *EMG data*, are not part of the top-level decision anymore. As such, they should be discarded from the decision requirements diagram. The updated decision model is provided in Figure 9.6.

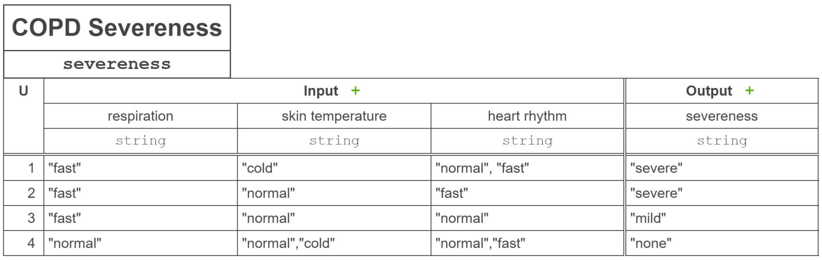
9.4.1.2 Decision input inclusion

$\Delta\Pi$ 2 (Decision input inclusion)**.** *A change ΔI_{d_+} indicates a change in the input set I_d of a decision D as follows: opposite to* $\Delta\Pi$*1, an input variable, or a new value for an existing variable, can be added to a decision table.*

χ 2 (Induced inconsistencies)**.** The process must provide the input data needed for decision enactment. If additional input data is required due to changes in the input set of a decision, the process is no longer capable of calling upon that decision.

Ψ **2** (Resolving inconsistencies)**.** The process needs to be redesigned to incorporate the new data that functions as part of the input for the decision. Thus, $I_{d^a} \subseteq I(a)$ must include the newly added input data when the decision activity $a \subseteq A_d$ calls upon decision D. This data must either be obtained from outside the process or produced within the process itself.

ε 2 (Example)**.** Notice that adding a new precondition variable results in the exact opposite changes to be applied on the process and decision models as the ones presented for $\Delta\Pi$*1*. If the precondition variable of *muscle activity* were to be added again to the decision table in Figure 9.4, the DRD should reintroduce the subdecision `Muscle activity` and its required *EMG data*. Furthermore, the process should be redesigned to be able to read the muscle activity sensor in order to provide the required input data to decision activity `Check COPD Severeness`, as to ensure the correct invocation of the decisions. Thus, this change pattern



COPD Severeness

severeness

U	Input +			Output +
	respiration	skin temperature	heart rhythm	severeness
	string	string	string	string
1	"fast"	"cold"	"normal", "fast"	"severe"
2	"fast"	"normal"	"fast"	"severe"
3	"fast"	"normal"	"normal"	"mild"
4	"normal"	"normal","cold"	"normal","fast"	"none"

Figure 9.4: Decision table with changed preconditions.

version would return the models to the initial state as shown in Figures 9.1, 9.2, and 9.3.

9.4.1.3 Decision output inclusion

ΔΠ 3 (Decision output inclusion)**.** *A change ΔO_{d_+} indicates a change in the output set O_d of a decision D as follows: a new output value can be added to a decision table.*

χ 3 (Induced inconsistencies)**.** The process must be able to interpret the decision outcomes for a sound process continuation. If the newly added decision outcomes impact the process control flow, the process should capture these outcomes in the control flow. Otherwise a deadlock occurs in the process.

Ψ 3 (Resolving inconsistencies)**.** The process may need to be redesigned to incorporate the newly added decision outcomes. Thus, if a decision impacts the control flow of the process, all $o_d \in O_{d^a} \subseteq O(a)$ must be captured in the process control flow following the decision activity $a \subseteq A_d$ that calls upon decision D.

ε 3 (Example)**.** Suppose that in the top-level decision table represented in Figure 9.2, the COPD severeness decision outcome of the first decision rule changes from *severe* to *lethal*. This new table is given in Figure 9.7 Notice that the outcome value *lethal* is a new outcome that was not represented in the decision table before. The process model in Figure 9.3 is not able to interpret this outcome in the decision point following the `Check COPD Severeness` decision activity. Hence the process model

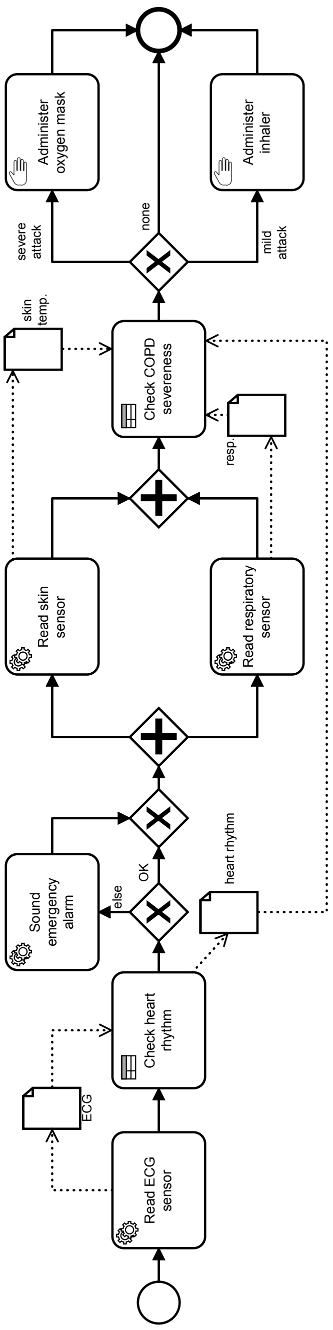


Figure 9.5: Process model adapted to changed preconditions.

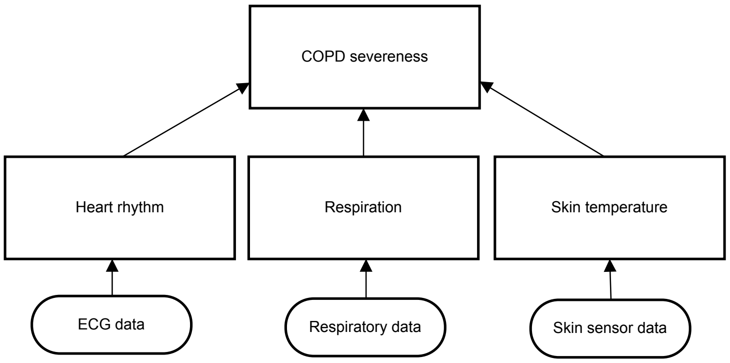


Figure 9.6: DRD adapted to changed preconditions in the top-level decision.

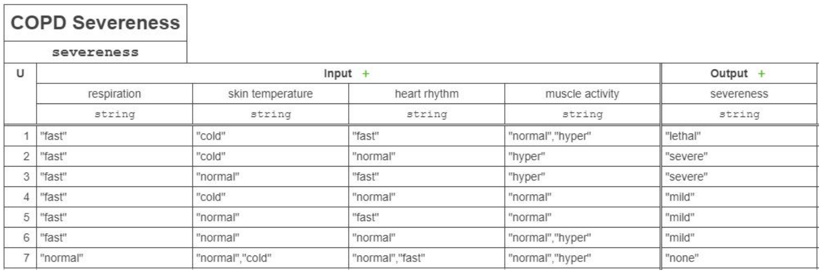
needs redesign to be able to capture that specific decision outcome as well, otherwise a deadlock can occur if the underlying `COPD Severeness` decision evaluates the patient condition to be *lethal*. The redesigned process, given in Figure 9.8, is able to interpret this outcome as well. Notice that in this case the DRD in Figure 9.1 does not undergo any change, since the changed decision outcome belongs to the top-level decision of the DRD. Hence, the outcome does not affect any decision constructs that are higher in the decision hierarchy.

9.4.1.4 Decision output exclusion

$\Delta\Pi$ 4 (Decision output exclusion)**.** *A change $\Delta O_{d\_}$ indicates a change in the output set O_d of a decision D as follows: an existing output value can be deleted from a decision table.*

χ 4 (Induced inconsistencies)**.** If a decision outcome that affects the control flow of the process is deleted from the decision model without redesigning the process model, dead branches, i.e., branches that used to capture the deleted decision outcome, may be introduced into the process.

Ψ 4 (Resolving inconsistencies)**.** An $o_d \in O_{d^a} \subseteq O(a)$ that is deleted from the decision table should result in eliminating its corresponding process branch or process branch conditions following the decision activity $a \subseteq A_d$.



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"	"lethal"
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 9.7: Decision table with changed outcomes.

***ε* 4** (Example)**.** The exact opposite changes to those in $\Delta\Pi 3$ occur in the models when this change pattern leads to deleting a decision outcome from the decision table. If in the decision table of Figure 9.7, the *lethal* outcome were again to be replaced by the *severe* outcome, then the decision table and process model would undergo redesign to revert back to the models presented in the running example, i.e., Figures 9.2 and 9.3.

9.4.1.5 Decision rule logic change

ΔΠ 5 (Decision rule logic change)**.** *A change ΔL indicates a change in the logic L of a decision D, i.e., a change in relating the existing input symbols I_d to the existing output symbols O_d within the decision table.*

***χ* 5** (Induced inconsistencies)**.** Given that the process and decision models were correctly integrated, a change in the underlying decision logic as such does not affect the consistency between the models. This is due to the fact that the process model and decision model are loosely coupled based on their input and output sets, as described in Definitions 9.3-9.5. Hence, the consistency between the models is dependent on these sets, and not on the underlying decision logic.

***Ψ* 5** (Resolving inconsistencies)**.** There is no need for process redesign, since changes in the underlying decision logic itself do not induce process and decision model inconsistencies.

***ε* 5** (Example)**.** Suppose that in the top-level decision table represented in Figure 9.2, the logic in the fourth decision rule

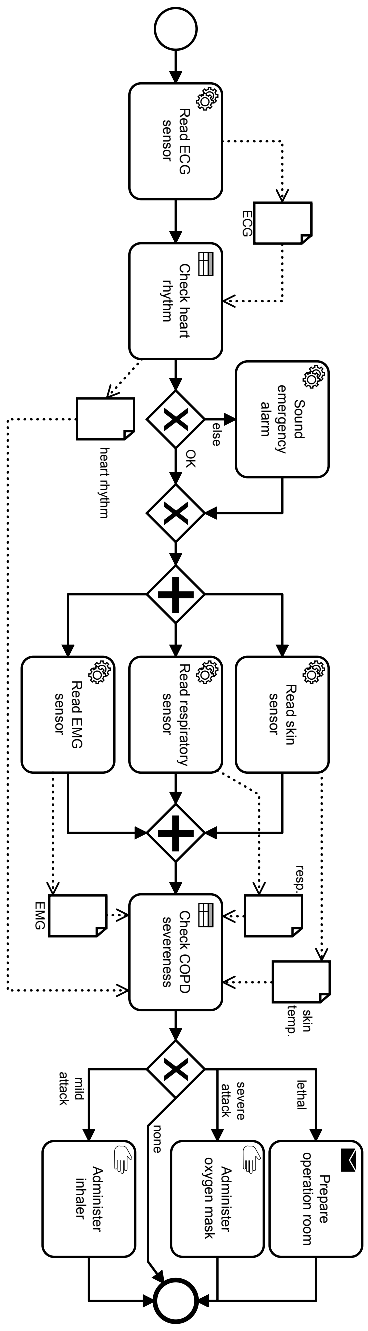
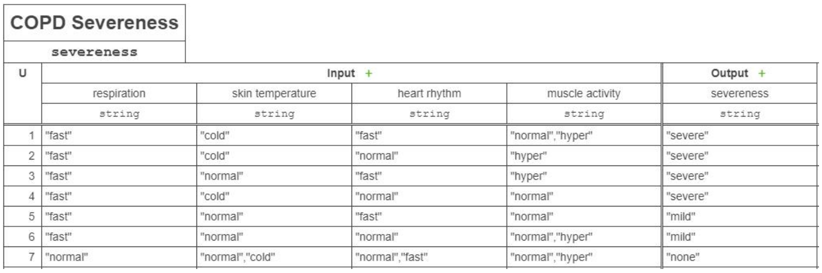


Figure 9.8: Process model adapted to changed outcomes.



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"	"severe"
5	"fast"	"normal"	"fast"	"normal"	"mild"
6	"fast"	"normal"	"normal"	"normal","hyper"	"mild"
7	"normal"	"normal","cold"	"normal","fast"	"normal","hyper"	"none"

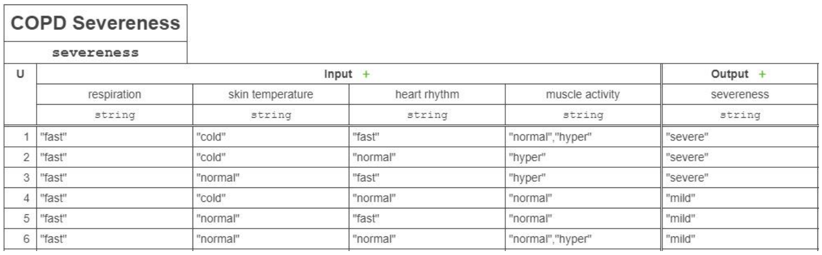
Figure 9.9: Decision table with changed logic.

changes. Instead of mapping the precondition values to an output of *mild*, the preconditions are now considered to be of a *severe* nature. This change is exemplified in the decision table in Figure 9.9. Note that this change does not affect the consistency between the process and decision models in Figure 9.3 and 9.1. This is due to the fact that only the underlying logic is altered by ΔΠ5, and not the input and output sets that form the loose coupling connections between the process and decision model [63, 66]. As such, the process in Figure 9.3 is still able to provide the decision model with the relevant input data, as well as to interpret the decision outcomes returned by the decision model. Thus, no process redesign is needed when this change pattern is applied.

9.4.1.6 Decision rule exclusion

ΔΠ 6 (Decision rule exclusion)**.** *If a decision rule* $i_d \in I_d \xrightarrow{L} o_d \in O_d$ *is deemed irrelevant at a certain point in time, it can be deleted in its entirety from a decision table.*

χ 6 (Induced inconsistencies)**.** This change pattern in essence triggers the same inconsistencies as three previously defined patterns, i.e., ΔΠ*1*, ΔΠ*4*, and ΔΠ*5*, since this change pattern is a combination of these individual patterns: it deletes the preconditions of a decision rule, the outcome of the decisions rule, as well as part of the logic that maps the preconditions to the outcomes. Hence, deriving from ΔΠ*1*, if the process still provides the combination of the deleted preconditions to the decision model, the decision model is not able to enact the requested



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"

Figure 9.10: Decision table with a deleted decision rule.

decision and inconsistency between the models arises. Likewise, deriving from $\Delta\Pi 4$, since a decision outcome was possibly deleted from the decision model, the process can possess dead branching conditions if the outcome for the deleted decision rule was a unique decision outcome within its decision table.

Ψ **6** (Resolving inconsistencies)**.** Resolving inconsistencies happens according to $\Delta\Pi 1, \Delta\Pi 4$, and $\Delta\Pi 5$ as well. On the one hand, an $o_d \in O_{d^a} \subseteq O(a)$ that is the outcome of the deleted decision rule should result in eliminating its corresponding process branch conditions following the decision activity $a \subseteq A_d$, if the deleted o_d is unique for its decision table. On the other hand, the process should make sure that the input set combination $i_d \in I_d$ is not provided to the decision model when decision activity $a \subseteq A_d$ calls upon decision D. This is due to the fact this that specific input combination has been deleted from the decision table and is thus not leading to a decision outcome that can be returned to the process for further interpretation. Thus, resolving these inconsistencies requires to redesign the process to fit both the input and output sides of the requested decision table excluding the deleted decision rule.

ε **6** (Example)**.** Suppose that decision rule 7 is entirely deleted from the decision table in Figure 9.2, rendering the decision table depicted in Figure 9.10. Notice that the decision table does not anymore contain the output *none*. However, the process model in Figure 9.3 does contain a control flow branch that captures this output of *none*. As a consequence, this branch in the control flow becomes a dead branch, since the decision outcome of *none* will never be reached in the `Check COPD severeness` decision activity. As such, this dead brach should be discarded from the process, as is the case in the redesigned process in Figure 9.11.

Notice, however, that by deleting the decision rule, the decision table is not complete anymore, i.e., the combination of precondition values that was present in rule 7 in Figure 9.2 can still manifest itself. However, the decision table is not able to return an outcome for this precondition combination. This could lead to a deadlock in the process. To avoid this, either the decision table should be completed and the preconditions values at hand should be mapped to other existing decision outcomes, or the process should be redesigned to capture the possibility of no decision outcome being returned. In Figure 9.11 this is achieved by terminating the process if the decision activity `Check COPD severeness` is not able to retrieve a decision outcome from the process model.

9.4.1.7 Decision rule inclusion

ΔΠ 7 (Decision rule inclusion)**.** *If a new decision rule $i_d \in I_d \xrightarrow{L} o_d \in O_d$ is deemed relevant at a certain point in time, it can be added in its entirety to an existing decision table.*

χ 7 (Induced inconsistencies)**.** Again, in analogy with $\Delta\Pi 6$, this change pattern is a combination of $\Delta\Pi 2$, $\Delta\Pi 3$, and $\Delta\Pi 5$: it adds both the preconditions and the outcome of a new decisions rule, as well as part of the logic that maps the preconditions to the outcomes. Hence, deriving from $\Delta\Pi 2$, if the process is unable to provide the newly added combination of preconditions to the decision model, the decision model will never be able to trigger the newly included decision rule. Likewise, deriving from $\Delta\Pi 3$, since a possibly new decision outcome was added to the decision model, the current process can possibly not capture that outcome, and thus, the process continuation is impaired if the new rule is triggered.

Ψ 7 (Resolving inconsistencies)**.** Resolving inconsistencies happens according to $\Delta\Pi 2$, $\Delta\Pi 3$, and $\Delta\Pi 5$ as well. On the one hand, an $o_d \in O_{d^a} \subseteq O(a)$ that is the outcome of the added

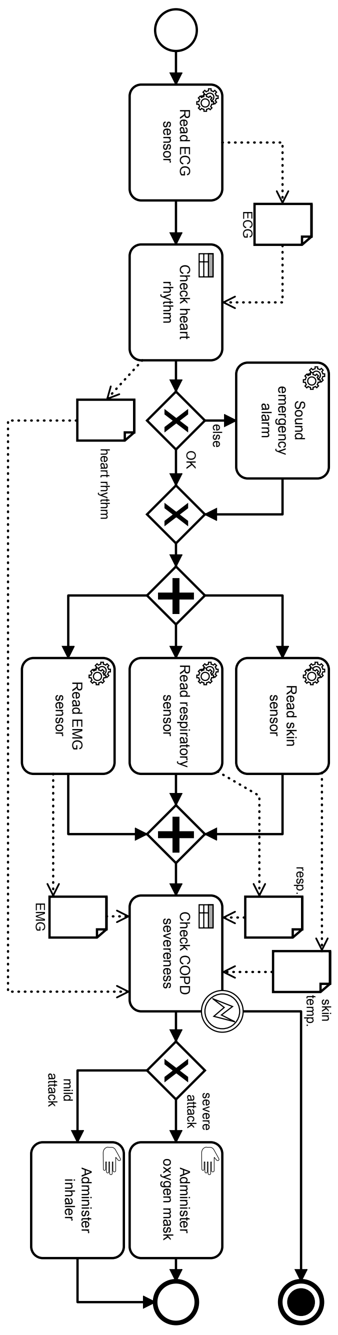


Figure 9.11: Process model adapted to exception handle deleted/missing rules.

decision rule should result in additional corresponding process branch conditions following the decision activity $a \subseteq A_d$. On the other hand, the process should make sure that the input set combination $i_d \in I_d$ is provided to the decision model when decision activity $a \subseteq A_d$ calls upon decision D. This is due to the fact this that specific input combination has been added to the decision table and is leading to a decision outcome that can be returned to the process for further interpretation. Thus, resolving these inconsistencies requires to redesign the process to fit both the input and output sides of the requested decision table including the added decision rule.

ε 7 (Example)**.** Suppose that the decision rule deleted in the previous change pattern is reintroduced again to the decision table. Thus, to the decision table in Figure 9.10, one rule is added, rendering the decision table in Figure 9.2. Note that the changes that need to be applied here are opposite to the ones discussed in the previous subsection. As such, the original process model in Figure 9.3 is obtained by redesigning the process model in Figure 9.11.

9.4.2 Decision requirements diagram change patterns

In the previous subsection, we have investigated changes localised within a single decision table. However, entire elements can be added or deleted from the DRD model as well. Definition 9.1 defines the elements of the DRD that can be added or deleted, i.e., decision nodes and input data nodes. By deleting the input data and decision nodes from the DRD, the information requirement arrows that connect them are deleted as well.

9.4.2.1 Decision node exclusion

$\Delta\Pi$ 8 (Decision node exclusion)**.** *A decision node $D \in D_{dm}$ can be deleted from the DRD. This corresponds to deleting all decision rules ($I_d \xrightarrow{L} O_d$) from a decision node D. Hence, this change pattern is an aggregation of multiple $\Delta\Pi 6$ changes. Note* *that deleting a decision node D also deletes all its incoming and outgoing information requirements arrows from the set IR.*

$\boldsymbol{\chi}$ **8** (Induced inconsistencies)**.** Since deleting an entire decision node corresponds to deleting all the individual rules encapsulated in that decision node, i.e., $\Delta\Pi 6$, the induced inconsistencies are the same as in $\chi 6$.

Ψ **8** (Resolving inconsistencies)**.** As multiple $\Delta\Pi 6$ change patterns result in multiple $\chi 6$ inconsistencies, they are all remedied according to $\Psi 6$ as well.

$\boldsymbol{\varepsilon}$ **8** (Example)**.** Suppose that the subdecision `Muscle activity` in the DRD in Figure 9.1 is excluded from the model. This also deletes the subdecision's input data, i.e., *EMG data*. The updated decision model is then given in Figure 9.6. The deleted `Muscle activity` subdecision was required by the top-level `COPD severeness` decision. Hence, the `Muscle activity` subdecision provided it's decision output as a precondition to the decision table underlying the `COPD severeness` decision. Since the `Muscle activity` subdecision is deleted, the preconditions of the higher-level decisions requiring the subdecision should be deleted as well, rendering a new COPD decision table presented in Figure 9.4. The process from Figure 9.3 needs to be adapted as well to reflect the changes, leading to the process depicted in Figure 9.5.

9.4.2.2 Decision node inclusion

$\boldsymbol{\Delta\Pi}$ **9** (Decision node inclusion)**.** *A new decision node D can be added to the set of decision nodes D_{dm}. This corresponds to adding multiple decision rules ($I_d \xrightarrow{L} O_d$) to a new decision node D. Hence, this change pattern is essence an aggregation of multiple $\Delta\Pi 7$ changes. Note that adding a decision node D also adds the necessary incoming and outgoing information requirements arrows to the set IR.*

$\boldsymbol{\chi}$ **9** (Induced inconsistencies)**.** Since adding an entire decision node corresponds to adding all the individual rules encapsulated in that decision point, i.e., $\Delta\Pi 7$, the induced inconsistencies are the same as in $\chi 7$.

Ψ **9** (Resolving inconsistencies)**.** As multiple $\Delta\Pi$*7* change patterns result in multiple χ*7* inconsistencies, they are all remedied according to Ψ*7* as well.

ε **9** (Example)**.** Suppose that a decision node `Muscle activity` is added to the the decision requirements diagram in Figure 9.6, effectively producing the DRD presented in Figure 9.1, as the *EMG data* input is added to the DRD as well. Since the top-level `COPD severeness` decision requires the `Muscle activity` subdecision, the preconditions in the `COPD severeness` decision table should reflect the outcomes provided by the `Muscle activity` decision table. As such, these preconditions are added to the top-level decision table, transforming the decision table in Figure 9.4 to the decision table provided in Figure 9.2. Likewise, the process needs to be adapted as well to reflect the changes applied to the decision model. As such, the process in Figure 9.5 is redesigned to become the process depicted in Figure 9.3.

9.4.2.3 Input data node inclusion

ΔΠ 10 (Input data node inclusion)**.** *A new data input node can be added to the set of data input nodes ID. By adding a data input node, its necessary input requirement arrows are also added to the set of IR and connected to the relevant decision nodes in D_{dm}. 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.*

χ **10** (Induced inconsistencies)**.** Given the equivalence with $\Delta\Pi$*2*, χ*2* inconsistencies apply. The legacy process is no longer capable of calling upon the decision, as it does not provide the necessary input data, given the addition of a new input data node.

Ψ **10** (Resolving inconsistencies)**.** As in Ψ*2*, $I_{d^a} \subseteq I(a)$ must include the newly added input data when the decision activity $a \subseteq A_d$ calls upon decision D. This data must either be obtained from outside the process or produced within the process itself.

ε **10** (Example)**.** Consider the decision model provided in Figure 9.6 with the top-level decision table in Figure 9.4. Assume that

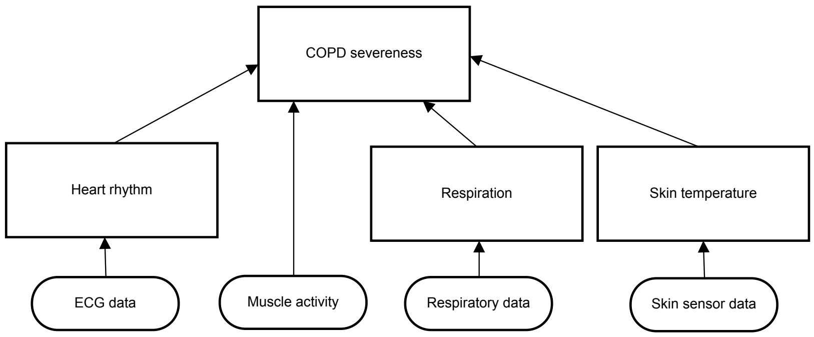


Figure 9.12: A DRD with an added input data node.

an additional input data node is added that provides information on muscle activity. The new decision model is then given in Figure 9.12, with the decision table in Figure 9.2 representing the underlying logic of the top-level decision. The process model in Figure 9.5 needs to undergo change to include the newly added input data. The process given in Figure 9.13 does exactly this. The `Perform muscle check` user task is carried out by a physician and the muscle activity is decided on by the physician and is input into the system such that the top-level decision `COPD severeness` can be evaluated.

9.4.2.4 Input data node exclusion

ΔΠ 11 (Input data node exclusion)**.** *A data input node can be deleted from the set of data input nodes ID. By deleting a data input node, all its input requirement arrows are also deleted from to the set of IR. 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.*

χ 11 (Induced inconsistencies)**.** Given the equivalence with $\Delta\Pi$*1*, χ*1* inconsistencies apply. The legacy process does no longer need to provide the deleted input data.

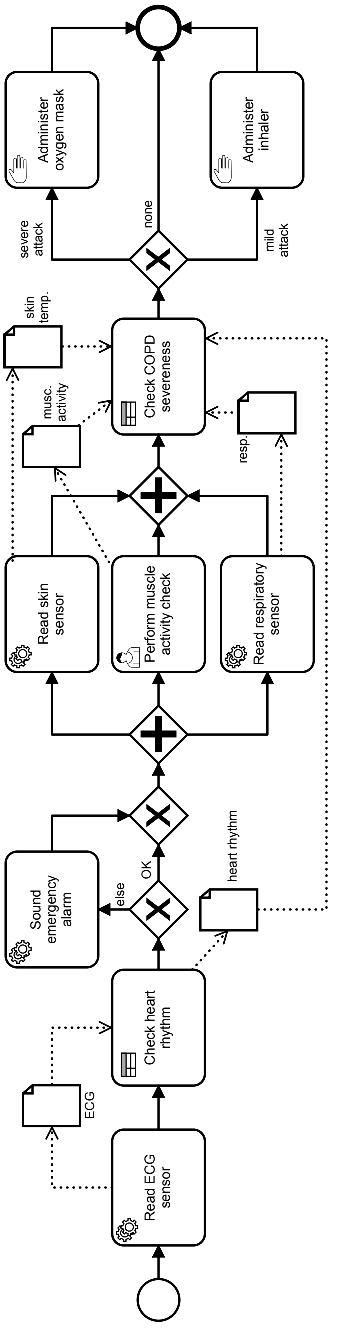


Figure 9.13: Process model adapted to manage the input data related to an added input data node in the DRD.

Ψ **11** (Resolving inconsistencies)**.** As in $\Psi 1$, $I_{d^a} \subseteq I(a)$ does not longer need to include the newly added input data when the decision activity $a \subseteq A_d$ calls upon decision D.

ε **11** (Example)**.** Consider the decision model provided in Figure 9.12 with the top-level decision table in Figure 9.2. Assume that the `Muscle activity` input data node is deleted from the requirements diagram since this information is now deemed invalid to make the top-level `COPD severeness` decision. The newly obtained DRD is then given in 9.6 with Figure 9.4 as the corresponding decision table. Again, the process needs to undergo redesign to adapt to the newly established decision model. As such, the process in Figure 9.13 is transformed into the process in Figure 9.5.

9.5 Change Propagation

In the previous section we have only assessed the direct effects of a change pattern in the decision model on the process and decision consistency. However, a single decision model change pattern can invoke a chain of additional change patterns in the decision model in order to ensure within-model consistency in the decision model, and consequently, may also have indirect additional effects on between-model consistency. For every change applied to the decision model, and for every additional change in the decision model triggered by the initial change, the process and decision consistency needs to be checked and maintained. Therefore, it is of paramount importance to assess the propagation of changes within the decision model. In what follows, we respectively discuss change propagation within the decision model and from the decision model to the process model.

9.5.1 Change propagation within the decision model

Change propagation throughout a model is an important aspect in hierarchical structures where referential integrity needs to be upheld. Typically, such structures are connected by tables, such as the relational database structure, where changes are propagated

Table 9.2: An overview of decision model change patterns and their influence on the process model.

Change pattern ($\Delta\Pi$)	**Induced inconsistency** (χ)	**Resolving inconsistencies** (Ψ)
$\Delta\Pi$ *1*: decision input exclusion	χ*1*: Possibly redundant data objects in the process.	Ψ*1*: Redesign process to get rid of redundant data.
$\Delta\Pi$ *2*: Decision input inclusion	χ*2*: Possibly insufficient input data to enact the decision.	Ψ*2*: Redesign process to incorporate new input data.
$\Delta\Pi$ *3*: Decision outcome inclusion	χ*3*: New decision outcomes are not captured in the control flow.	Ψ*3*: Redesign process with new branch conditions.
$\Delta\Pi$ *4*: Decision outcome exclusion	χ*4*: Deleted outcomes can leave dead branches in the process flow.	Ψ*4*: Redesign process by deleting redundant branch conditions.
$\Delta\Pi$ *5*: Decision rule logic change	χ*5*: Processes and decisions are loosely coupled through input and output sets. Thus, no inconsistencies arise by changing the mapping.	Ψ*5*: No need for process redesign as there is no change in the input and output sets.
$\Delta\Pi$ *6*: Decision rule exclusion	χ*6*: This pattern is a combination of $\Delta\Pi$*1*, $\Delta\Pi$*4*, and $\Delta\Pi$*5*. Hence, the same inconsistencies arise here as well, i.e., χ*1* and χ*4*.	Ψ*6*: In process: delete redundant data (Ψ*1*) and delete redundant conditions (Ψ*4*).
$\Delta\Pi$ *7*: Decision rule inclusion	χ*7*: Again, this pattern is a combination of $\Delta\Pi$*2*, $\Delta\Pi$*3*, and $\Delta\Pi$*5* and the same inconsistencies arise here as well, i.e., χ*2* and χ*3*.	Ψ*7*: In process: add additional data (Ψ*2*) and add new branching conditions (Ψ*3*).
$\Delta\Pi$ *8*: Decision node exclusion	χ*8*: This pattern is an aggregation of multiple $\Delta\Pi$*6* changes. Hence, the same inconsistencies apply again, i.e., χ*1* and χ*4*.	Ψ*8*: In process: delete redundant data (Ψ*1*) and delete redundant conditions (Ψ*4*).
$\Delta\Pi$ *9*: Decision node inclusion	χ*9*: This pattern is an aggregation of multiple $\Delta\Pi$*7* changes. Hence, the same inconsistencies apply again, i.e., χ*2* and χ*3*.	Ψ*9*: In process: add additional data (Ψ*2*) and add new branching conditions (Ψ*3*).
$\Delta\Pi$ *10*: Input node inclusion	χ*10*: Given the equivalence with $\Delta\Pi$*2*, χ*2* inconsistencies apply.	Ψ*10*: In process: add additional data (Ψ*2*).
$\Delta\Pi$ *11*: Input node exclusion	χ*11*: Given the equivalence with $\Delta\Pi$*1*, χ*1* inconsistencies apply.	Ψ*11*: In process: delete redundant data (Ψ*1*).

by rules such as on delete cascade or on update cascade [27, 107]. Decision tables that are used in decision models are of a similar hierarchical structure, and hence, similar change propagation can be expected. However, instead of cascading an update or delete action throughout the whole model, the change propagation can also be captured by applying other change patterns.

When applying one of the change patterns presented in Table 9.1, the referential integrity of the DMN model in its entirety needs to be upheld. As such, for every change pattern performed, the effects on the decision hierarchy need to be assessed. This is done by both taking into account the propagation to the higher as well as lower levels of the DMN decision model hierarchy:

- If a change pattern deletes an element from the DMN decision model:
  - The higher-level elements that are potentially requiring the deleted element can render the decision model inconsistent as the deleted information is no longer provided. Resolving this issue requires additional change patterns that:
    * Either delete the higher-level elements that are not provided with the required information.
    * Or introduce new lower-level elements that provided the required information.
  - The lower-level elements that are potentially providing information concerning the deleted element can render the decision model inconsistent, as the provided information is no longer captured. Resolving this issue requires additional change patterns that:
    * Either delete the lower-level elements whose information is no longer used higher up in the decision hierarchy.
    * Or introduce new higher-level elements that capture the information provided by the lower-level elements.
- If a change pattern adds an element to the DMN decision model:

- The higher-level elements are potentially receiving information that they can not capture adequately. As such, the newly added element renders the decision model inconsistent. Resolving this issue requires additional change patterns that:
  - Either introduce higher-level elements that capture the information provided by the newly added element.
  - Or delete the newly added element to restore the within-model consistency.
- The lower-level elements are potentially not providing the adequate information to the newly added element. As such, the newly added element renders the decision model inconsistent. Resolving this issue requires additional change patterns that:
  - Either introduce lower-level elements that provide the required information to the newly added element.
  - Or delete the newly added element to restore the within-model consistency.

As such, resolving within-model inconsistencies requires the administration of additional change patterns that on their turn either exclude elements from the decision model or include new elements to the decision model. These additionally triggered change patterns can in turn trigger yet other change patterns. Thus, a chain of triggered patterns may need to be propagated through the decision model until the system stabilises. For instance, when pattern $\Delta\Pi 1$ is applied, $\Delta\Pi 8$ may need to be triggered. Change pattern $\Delta\Pi 8$ can in turn trigger $\Delta\Pi 11$ after which the propagation of changes stops and the system returns to a stable state.

9.5.2 Change propagation to the process model

This subsection deals with change propagation from the decision model to the process model. Applying a change pattern to the decision model can induce inconsistencies in the integration of the process and decision models. Hence, when a decision model change pattern is applied, one must assess its effects on the process model, i.e., whether the decision model change pattern propagates changes to the process model as well. Notice that only change patterns that impact the inputs and/or outputs of a decision used in the process model are relevant for change propagation to the process model. This is due to the fact that consistently integrated process and decision models view decisions as services which provide decision outputs after being invoked by the process that presents the relevant input data [63, 66]. If the change patterns applied to the decision model provoke a change to the process model for consistency reasons, changes need to be applied to the process model to restore consistency.

In the previous sections, we have demonstrated which change patterns can occur in a decision model, and how the change patterns generate inconsistencies. Furthermore, for every specified change pattern, we have demonstrated how to redesign the process to restore consistency between the process model and its underlying decision model. Notice that, as specified in Table 9.2, for every change pattern-induced process and decision model inconsistency, the problem can be threefold: either the process is not providing the relevant data as input to the decision model due to the applied change pattern; or the process is not able to interpret the decision result returned by the decision model due to the applied change pattern; or both the inputs provided by the process to the decision model and the interpretation of the decision outputs in the process provide difficulties. Thus, after applying decision model change patterns, these inconsistencies between the process and decision models can arise. Resolving these inconsistencies is achieved, on the one hand, through ensuring that the interfacing between the process and the decision model inputs is correct; and on the other hand, through safeguarding the interfacing between the decision model outputs and the process. Hence, in those cases, changes need to be applied to the process model to restore the between-model consistency.

9.5.3 Resolving decision input inconsistencies

Notice that a process and decision model consistency implies that for each (sub)decision invoked in the process it must be guaranteed that the (sub)decision is invocable at that stage of the process

[66]. In order to invoke a certain decision in the process through a decision activity, all relevant data needed for invoking that decision and its subdecisions must be available. Additionally, if subdecisions of the decision that is invoked are modelled within the process by means of decision activities, the intermediate results of the subdecisions must be readily available as well. Violations against the invocability of decisions from the process can be resolved in three simple steps [66]:

1. Apply the topological hierarchy of decisions from the decision model to the order of modelled decision activities in the process.

2. Ensure that all input data needed in the decision model is present in the process model as well, either as external data or as internally produced process data.

3. For every decision activity in the process make sure that all input data and intermediate results of its subdecision activities in the process are linked to the decision activity as input data objects.

Thus, applying these changes (adding all relevant input data or subdecision outcomes in the hierarchical order) to the process will ensure that the process model is aligned with the decision model inputs.

9.5.4 Resolving decision output inconsistencies

The previously enumerated process changes solve the process and decision model inconsistencies from the input perspective. However, the process still needs to correctly interpret the decision outcomes after the decision has been enacted [10, 67]. Hence, the process needs to catch the relevant decision outcomes in its control flow and to disregard irrelevant decision outcomes that are not possible anymore due to decision model changes. Note that the decision outcomes do not necessarily influence the control flow perspective of the process. Often, the decision outcome influences the data perspective of the process and is used as a data object that can be consumed by another activity later on in the process. For instance, the outcome of a subdecision can be incorporated as a data object in the process, which on its turn can be consumed by a decision activity which invokes a higher-level decision that requires the outcome of the subdecision. Likewise, decision outcome data objects can be used as input for operational activities or automated tasks.

Hence, applying these changes (adding all relevant output objects, branches, and conditions needed to capture decision outcomes) to the process will ensure that the process model is aligned with the decision model outputs.

9.6 A Proof-of-Concept Modelling Environment

In this section we present the capabilities of the proof-of-concept modelling environment we have developed for providing modelling support for decision-aware business process evolution. We describe the capabilities of the tool and illustrate how the modelling environment works through an example.

9.6.1 Description of the modelling environment

The described approach for decision-aware process evolution was implemented as a modelling environment to interactively guide modellers in change propagation and model evolution. The modelling environment exists out of two components:

1. A first component checks the within-model consistency of the DMN decision model after one of the change patterns from Table 9.1 is applied[17]. After discovering errors in the model, the tool suggests actions which can be taken to resolve the errors. By selecting one of the actions, the tool automatically performs the action and checks for within-model inconsistencies again. As such, the model can be evolved consistently in an iterative fashion.

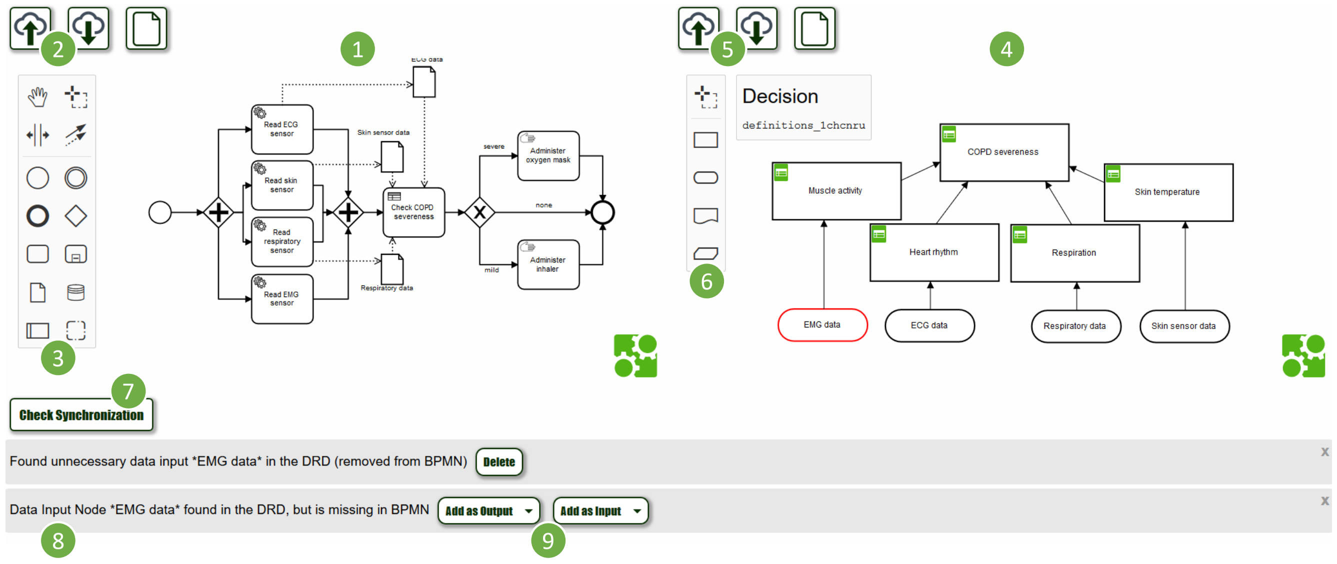
[17]The first component (within-model checking) can be accessed at `http://dmn.fg-bks.uni-koblenz.de/?dmnurl=dmn/BaselineModel.dmn`. This component was described in the previous two chapters.

2. The second component checks for between-model consistency, i.e., the consistent integration between a DMN decision model and a BPMN process model[18]. Here too, the modelling environment proposes concrete actions that the user can chose to resolve possible inconsistencies. Consequently, the system will automatically perform the actions.

Figure 9.14 shows the interface of the BPMN + DMN verification in the modelling environment, with numbers indicating the different parts of the tool, which are as follows:

1. Display of the BPMN process model.
2. Buttons to upload a BPMN model into the environment, to download the model from the environment, and to generate an empty BPMN model.
3. BPMN elements which can be selected by the modeller to add to the BPMN model.
4. Display of the DMN model. By clicking on one of the decision nodes, the decision table view can be accessed. In that view, a *View DRD* button can bring the modeller back to the DRD view.
5. Buttons to upload a DMN model into the environment, to download the model from the environment, and to generate an empty DMN model.
6. DMN elements which can be selected by the modeller to add to the DMN model.
7. The *Check synchronization* button. By clicking this button, the modelling environment checks wether the BPMN and DMN models are consistenly integrated, i.e., whether there are errors which need to be corrected.
8. Feedback for the modeller that explains the errors. Here, the error messages as a result of the verification mechanism between the DMN and BPMN models of the modelling environment are displayed and explained.

[18]The second component (between-model checking) can be accessed at `http://dmn.fg-bks.uni-koblenz.de/bpmn-verifier/index.html`



Found unnecessary data input *EMG data* in the DRD (removed from BPMN)

Data Input Node *EMG data* found in the DRD, but is missing in BPMN

Figure 9.14: The BPMN + DMN verification view of the modelling environment

9. Action buttons to resolve the errors. Next to the error messages, automatic actions are suggested by the modelling environment that can resolve the errors. By clicking any of the suggested action buttons, the modelling environment automatically performs the selected action to the BPMN or DMN model.

The core of the environment is the detection of inconsistencies, either within the decision model, or between the decision model and the process model. Means to detect the inconsistencies are denoted as individual verification capabilities. The work in [40] already presents a small survey of DMN decision logic level capabilities supported by existing tools. For this work, the survey was extended in order to classify existing approaches with regard to the new verification capabilities discussed in this work. Table 9.3 shows an overview of supported verification capabilities by existing approaches for analysing DMN models on the decision logic level as well as the decision requirements level. Additionally, the table shows between-model verification capabilities for DMN and BPMN models. As can be derived from Table 9.3, most work has focused on decision logic verification capabilities, i.e., checking for inconsistencies within a decision table. However, decision logic verification between linked decision tables, decision requirements hierarchy verification, and process-decision verification capabilities are still underrepresented in the literature. Hence, in our proof-of-concept modelling environment, we also include these neglected verification capabilities. The verification capabilities mentioned in Table 9.3 are in part based on the classifications from [40, 124]. In what follows, we briefly describe all the verification capabilities mentioned in Table 9.3[19].

Overview of verification capabilities covered by existing approaches (X = full and o = partial support).

9.6.1.1 BPMN + DMN verification capabilities

– **Unused Data Object Reference verification (BPMN).** Detecting data object references in the BPMN model which are missing in the DRD.

[19]For the description of the decision logic and DRD verification capabilities, we refer to the previous chapter.

Table 9.3: Overview of verification capability coverage (X = full and o = partial support).

		Decision logic capabilities											**DRD level capabilities**						**BPMN + DMN**					
Literature		Identical Rules	Equivalent Rules	Subsumed Rules	Indeterminism	Overlapping Conditions	Partial Reduction	Missing Rules	Unused Predefined Value	Missing Predefined Value	Missing input value	Missing output value	Missing input column	Missing output column	Idle data input	Missing (data) input	Multiple (data) input	Inconsistent types	Unused Data Object Reference	Missing Data Object Reference	Unused Data Input	Missing Data Input	Unused Table Output	Missing Table Output
Calvanese et al. (2016)	[29]	X	o	X		X		X																
Laurson et al. (2016)	[91]	X		X	o	X	o	X																
Batoulis et al. (2017)	[8]	X		X		X		X															X	X
Calvanese et al. (2017)	[30]	X	o	X	o	X		X																
Ochoa et al. (2017)	[104]										X	X												
Batoulis et al. (2018)	[7]	X		X		X		o																
Calvanese et al. (2018)	[31]	X	o	X	o	X	o	X																
Corea et al. (2018)*	[39]	X			X	X																		
Corea et al. (2019)*	[40]	X	X	X	X	X	X	X																
This work		**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**

- **Missing Data Object Reference verification (BPMN).** Detecting missing data object references in the BPMN model which are present in the DRD.
- **Unused Data Input verification (DMN).** Detecting data input nodes in the DMN model which are missing in the BPMN model.
- **Missing Data Input verification (DMN).** Detecting missing data input nodes in the DMN model which are present in the BPMN model.
- **Unused table output verification.** Detecting if there is a rule output which is not referenced in the BPMN model (either as an edge after the rule task or a data output).
- **Missing table output verification.** Detecting if there is an option in the BPMN, e.g. an outgoing edge, but no corresponding rule output (i.e., that path in the BPMN is unreachable).

The modelling environment is based on the open source Camunda[20] modeller which we have advanced with verification capabilities. The modeller can apply any of the change patterns discussed in this paper. After doing so, the modelling environment checks for consistency errors and displays error messages. The modelling environment can highlight the errors and suggests actions to remedy the inconsistencies. After the modeller selects an action, the modelling environment automatically performs it and checks for errors again. This way, the DMN can be evolved iteratively in a consistent way and the consistency with the BPMN model can be checked.

9.6.2 Example of model evolution as supported by the modelling environment

In this subsection we provide a short example to show how, respectively, the DMN evolution and verification, and the BPMN +

[20]https://camunda.com/download/modeler/

DMN evolution and verification work[21]. We would like to remind the reader that the modelling environment can be accessed online (see link in Section 7.1).

9.6.2.1 BPMN + DMN model evolution and verification

Suppose that a process model is built which makes use of the initial decision model presented in Figure 9.1. In the BPMN + DMN view presented in Figure 9.15, the two models are visible next to each other. In what follows, we illustrate some of the BPMN + DMN verification capabilities of the modelling environment, which were mentioned in Table 9.3.

Suppose that the *EMG data* object is deleted from the BPMN model in Figure 9.15, i.e., one of the necessary decision inputs for the `COPD severeness` decision is deleted. After checking for the synchronisation between the BPMN and DMN models, the modelling environment displays and highlights errors as shown in Figure 9.16. The EMG input data node in the DRD is highlighted as it is not referenced by the process model. The modelling environment suggests actions which can be taken towards remedying the situation: either deleting the input data node from the decision model, or adding a corresponding data object to the BPMN process model. By selecting one of the BPMN process activities from one of the drop-down menus (*Add as input* or *Add as output*), a data object referencing the relevant input data node in the DRD is added to the selected activity as an input or output object respectively. By selecting *Add as input* to the `Check COPD severeness` activity, the initial situation in Figure 9.15 is restored.

Suppose now that the input data node *EMG data* is deleted in the DRD model. As such, a data object in the BPMN model, i.e., *EMG data*, which serves as an input to the `Check COPD severeness` decision activity, does not have a corresponding input data node in the DRD model, as shown in Figure 9.17. The modelling environment suggests either to delete the data object from the BPMN process model or to add an input data node to one the decision nodes in the DRD. After selecting the latter, the

[21]For the DMN evolution and verification part, we refer to the previous two chapters.

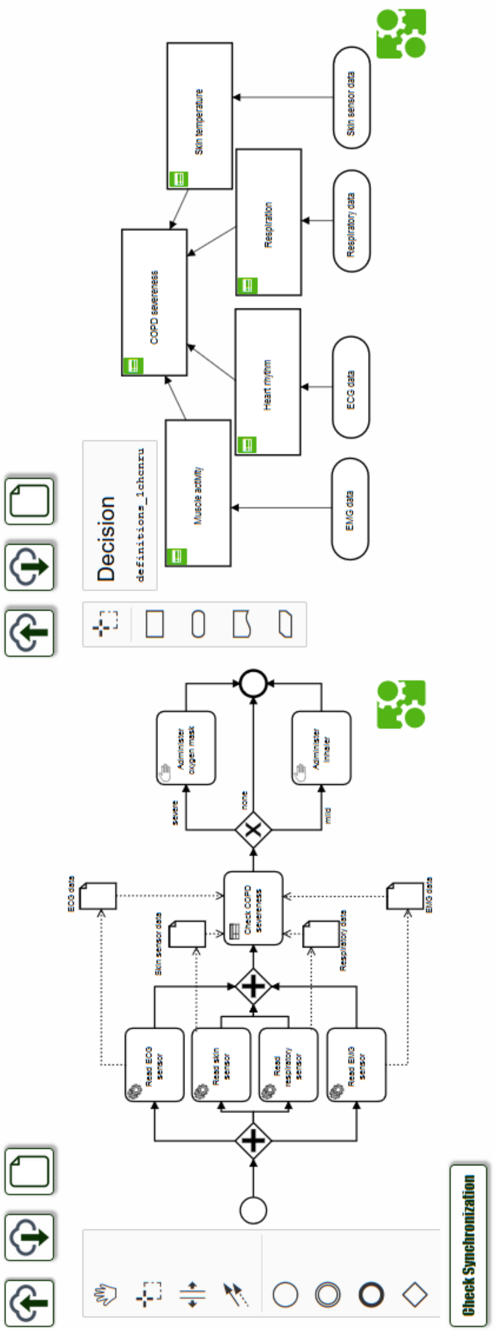
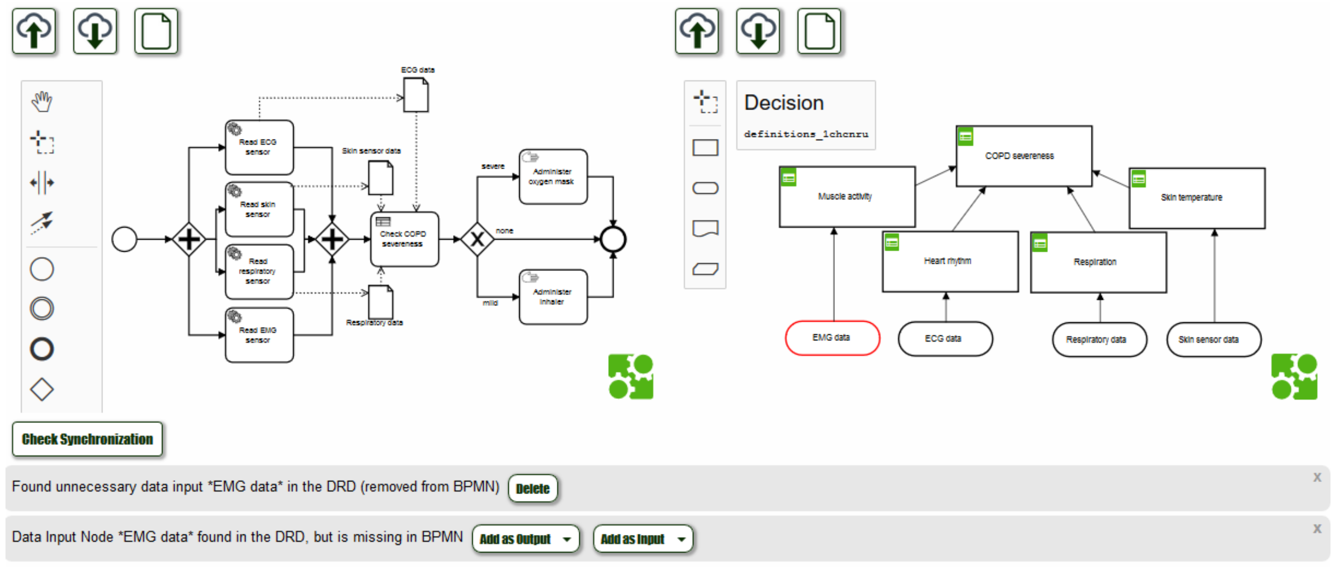


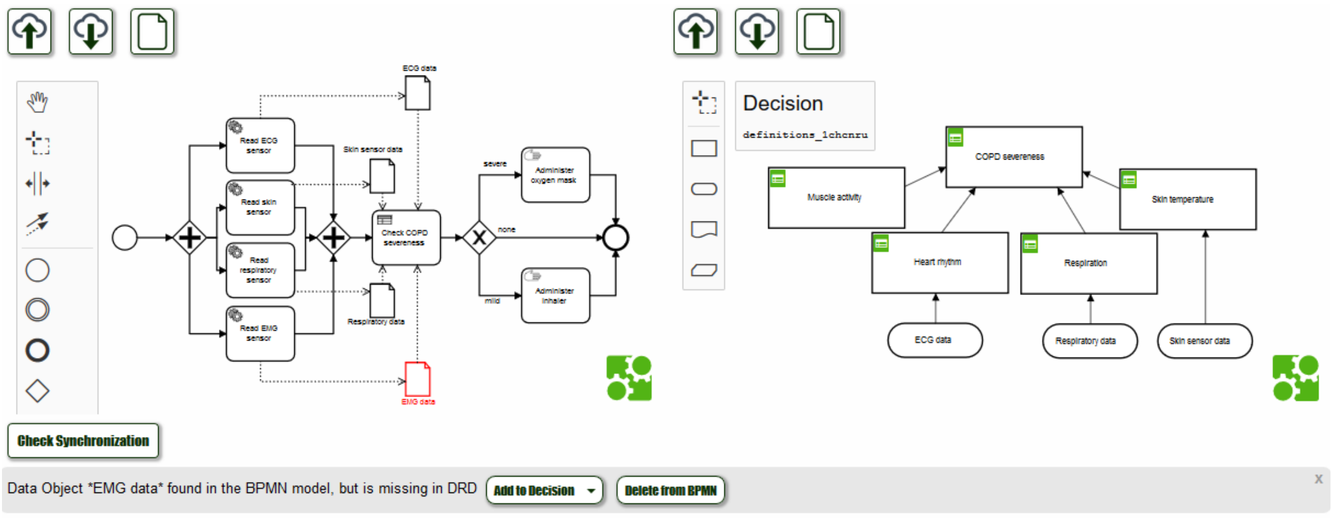
Figure 9.15: Initial view of the integrated BPMN and DMN models in the modelling environment.



Found unnecessary data input *EMG data* in the DRD (removed from BPMN)

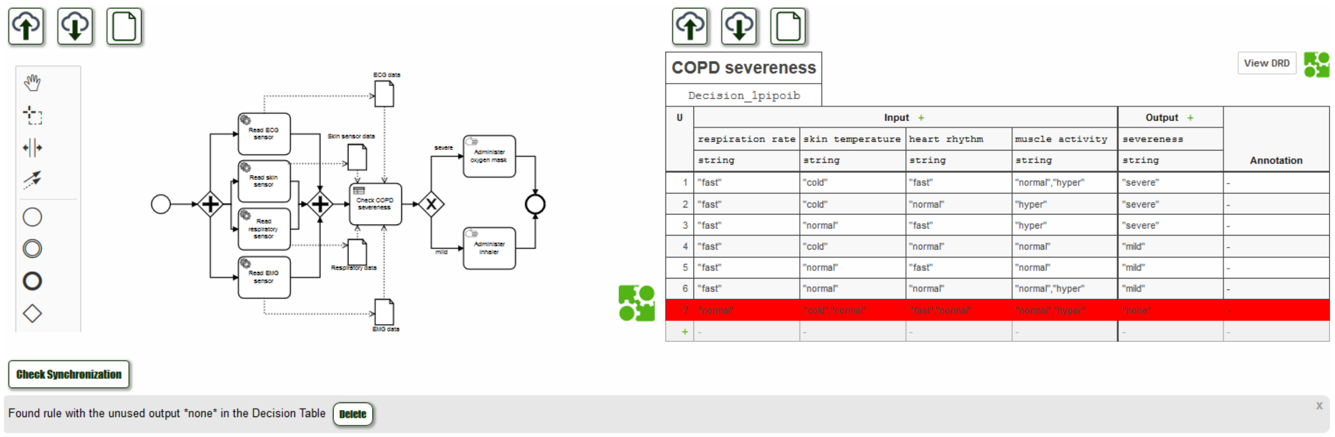
Data Input Node *EMG data* found in the DRD, but is missing in BPMN

Figure 9.16: Detecting a missing data object needed for decision input in the BPMN process model.



Data Object *EMG data* found in the BPMN model, but is missing in DRD

Figure 9.17: Detecting an input data node in the DRD model which is referenced by a data object in the BPMN model.



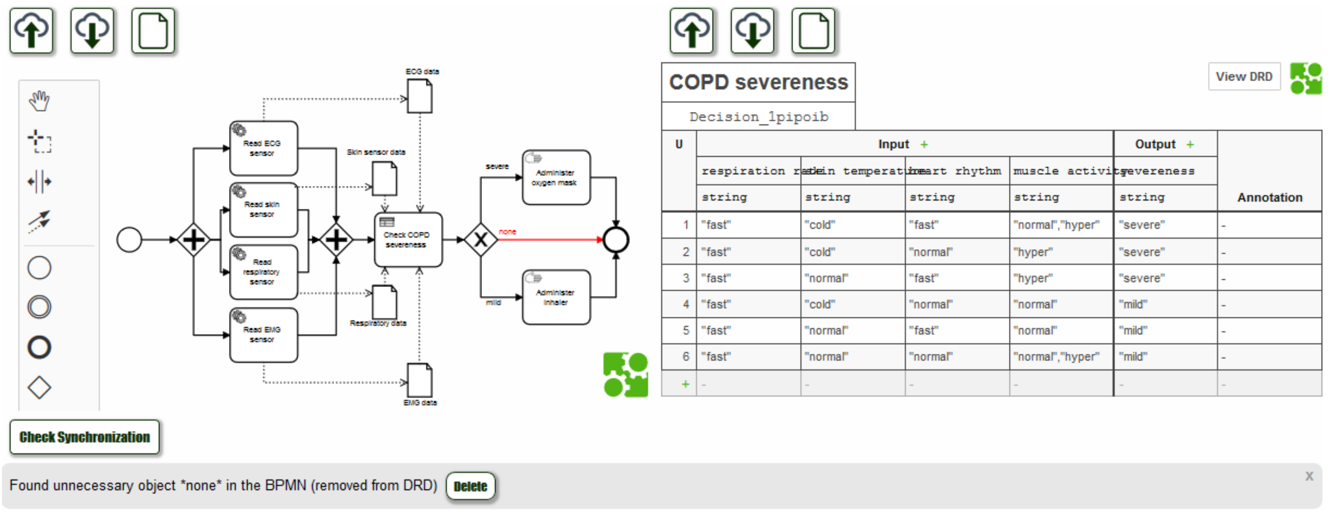
COPD severeness

Decision_1pipoib

U	Input				Output	Annotation
	respiration rate	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"	"cold","normal"	"fast","normal"	"normal","hyper"	"none"	-
+	-	-	-	-	-	-

Found rule with the unused output *none* in the Decision Table

Figure 9.18: Detecting a decision rule output which is not captured in the BPMN model.



COPD severeness

Decision_1pipoib

U	Input +				Output +	Annotation
	respiration r	skin temperat	heart rhythm	muscle activit	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"	-
+	-	-	-	-	-	-

Found unnecessary object *none* in the BPMN (removed from DRD) Delete

Figure 9.19: Detecting an unreachable path in the BPMN model (no corresponding decision output in the table).

initial situation in Figure 9.15 is restored again.

Suppose that in the initial situation in Figure 9.15 the edge which captures the `COPD severeness` decision outcome of *none* is deleted from the process model. Figure 9.18 shows this situation. The decision rule whose outcome is not captured in the BPMN process model is highlighted in the decision table view of the DMN model. The modeller can either delete the decision rule that is not captured in the process, or remodel the outgoing edge following the `Check COPD severeness` decision activity to recapture the neglected decision outcome. After opting for the latter, the initial situation in Figure 9.15 is obtained.

Suppose that, instead of deleting an edge that captures a decision rule outcome in the process model, the decision rule itself is deleted from the decision table while the edge remains intact, e.g., the highlighted decision rule 7 in Figure 9.18 is deleted. As such, the `COPD severeness` decision outcome of *none* is no longer available in the decision table, while a control flow edge in the BPMN model exists that tries to capture this decision outcome. Thus, this edge is no longer reachable in the BPMN process model. This situation is illustrated in Figure 9.19. The modeller can either decide to remodel the decision rules with the outcomes that are present in the non-covered edges. Alternatively, if the deleted decision rule is indeed deemed unnecessary, the idle edge itself can be deleted from the BPMN process model.

9.7 Conclusion and Future Work

This paper investigates change patterns in an integrated decision- and process modelling environment. As we have shown through a running example, changes in the decision model can lead to inconsistencies, either within the decision model itself, or between the decision model and the process model. Here, executing other change patterns to resolve these inconsistencies might be necessary. To this aim, decision model change patterns and their influence on the process model are defined in this work, allowing to propagate change patterns and thus to ensure consistent model evolution. The means presented in this work allow to pin-point those inconsistencies that can arise during modelling, and prescribe specific change patterns to promote flexible and sustainable development of both process and decision models in integrated modelling environments. Furthermore, we illustrate the feasibility of our approach by providing a modelling environment prototype, which can be used for consistent model evolution. The prototype implements inconsistency detection via so-called verification capabilities and suggests change patterns to resolve inconsistencies in a semi-automated manner.

In future work, we will evaluate our modelling environment prototype on industrial case-studies and usability tests. Here, we aim to further investigate the feasibility of our approach with real-life data sets, and to examine the cognitive effects of using the tool for human modellers, e.g., whether the tool allows to evolve models with higher efficiency, less errors, and less mental effort needed. Furthermore, we will investigate how extending DMN and BPMN with a shared ontology can be exploited to ensure consistency between process and decision models, as well as the use of DMN for context aggregation in processes [97]. Such a shared viewpoint could allow to incorporate external domain knowledge in order to further promote consistent model evolution. We will also look into evolving decision and process models at runtime, as it is necessary to evolve models without incurring costs that are related to shutting down the system to perform the required adaptations [121].

Part V

Epilogue

CHAPTER 10

Final Remarks

"Shvatam te:
Čovjek si u jednom prostoru i vremenu
Što živi tek sada i ovdje
I ne zna za bezgranični
Prostor vremena
U kojem se nalazim
Prisutan
Od dalekog jučer
Do dalekog sjutra
Misleći
O tebi
Ali to nije sve"

("I understand you:
You are a man in one space and time
Who lives only here and now
And knows nothing about the infinite
Space of time
Where I am
Present
From distant yesterday
Till distant tomorrow
Thinking
About you
But that's not all.")

Putovi
— *Mehmedalija Mak Dizdar*

This chapter provides assumptions, clarifications, and final remarks with regard to the contributions detailed in Parts II - IV.

10.1 Part II: Integrating Processes and Decisions

When moving from process models that contain hard-coded decision logic to the separation of modelling concerns and their subsequent consistent integration, we have opted for the merging and parametrisation of activities that are carried out as a result of a decision path. This was the case in the transition from Figure 2.3 to Figures 2.4 and 2.5 in Chapter 2, where the distinct activities for drawing up a contract were merged and parametrised into a singe `Draw up contract` activity. A similar approach was used in Chapter 4: in the transition from Figure 4.2 to Figure 4.3, the distinct cooling system activities that were used in Figure 4.2 to execute the conclusion of the decision path, were merged and parametrised into a single `Set cooling level` activity in Figure 4.3. Note that this approach moves part of the complexity from the visual process model to the parametrisation of the newly merged activity. Furthermore, it is worthwhile noticing that such an approach of activity merging and parametrisation cannot always be applied. Indeed, often there is a need to maintain distinct activities, notwithstanding their limited differences. This need for activity differentiation can for instance be motivated by resource allocation: despite the limited distinction between the activities, a different resource may need to be allocated to the specific activities. For instance, in Figure 2.3, the company's policy may be that low risk customer contracts are drawn up by the administrative staff, while high risk contracts are to be drawn up by the risk manager. In comparable cases, activity merging and parametrisation are not advised. Rather, this merging and parametrisation should be carried out in cases of decision automation or system automated tasks, i.e., in situations where changing resource allocation and handover of work are not issues that need to be considered for the specific activities.

In this thesis, we indeed mainly focused on modelling decisions that are crisp and explicit for automated decision making. However, modelling tacit decision knowledge of workers is often a major challenge. For instance, in Figure 3.3, we opted to model the `Appraise property` activity as a regular activity rather than a decision activity whose logic is stored in the underlying decision model, assuming that property appraisal is carried out by an expert, whose tacit knowledge and experience provide the expert with the ability to appraise the property by inspecting it. Making this knowledge and experience explicit by modelling it in a decision table format is a challenging endeavour, as many variables are in play, with each property being specific and with many possible exemptions and special cases.

Note that, unlike in related works as explained in Chapter 2, in this dissertation, we adopt a broad definition of a decision in a process, since we recognise that decisions can be modular and distributed across the entire process, rather than localised to a single decision point. We define a decision in a process as the transformation of inputs to outputs given a certain logic in a decision activity, as stated in Definition 2.9. The instances that pass through the activity carry all the data that are attached to that instance and not only the data relevant for the decision invocation. Therefore the input data for the decision I_{d^a} is a subset of all the data carried as input to the decision activity $I(a)$. The same holds for the output data leaving the activity. A subset, O_{d^a}, of the data that is leaving the activity $O(a)$, represents the outcome of the decision. As such, theoretically, a single decision activity can in fact invoke multiple decisions with each their own decision logic $L_{d^a} \subseteq L$. A simple example would be a decision table with two output columns (output sets). These would be considered two decisions in the process. In event logs it is possible that different logic is perceived in the same process activity, mapping different inputs to different outputs in that same activity (i.e., manifesting multiple decisions in the same activity). Given the narrow definitions of a decision in other works, we opted to define decisions as broadly as possible such that the definition of a decision in a process holds both for modelling and mining purposes.

10.2 Part III: Decision as a Service (DaaS)

In analogy with Part II, the decision model in this part is considered as given, correct, and static. This part deals with the interactions that processes can have with the given decision model. These interactions are determined by the possible decision services that can be derived from the decision model. As such, this part presents a decision-first approach, i.e., based on the decision model, for the interaction between processes in general and a decision model that they want to invoke. This decision-first approach is why the Figure 5.2 is oriented the way it is: the decision layer is at the top as we start from the decision model. The decision service layer is derived from the decision model in the decision layer: possible decision services that allow for the invocation of the underlying decision model, or parts of it, are constructed. Note that for the invocation of the same decision, different decision services can be used. For the sake of clear representation, Figure 5.2 only shows the decision services that are called upon by the process variant depicted in the process layer. Other process variants, or indeed, other processes, can use other decision services.

Whether a process will be able to correctly invoke a decision service will be determined by the Service Adherence Criterion in Definition 5.10. Note that a process can adhere to one decision service that invokes a decision, while not conforming to other decision services that pertain to the same decision in the decision model. As such, the integration criteria that are discussed in Part II are based on the interaction of a single process with a decision model. The DaaS design and the service adherence criterion imply that different process variants or indeed entirely different processes can interact with a decision model in multiple ways, as long as they adhere to one of the decision services that is capable of invoking their requested decision.

As mentioned earlier, a process can interact with the decision model in multiple ways, even when considering a single decision. To illustrate this consider the top-level decision in the decision model in Figure 10.1, i.e., decision `A`. Multiple decision services, and thus multiple ways of invoking the decision logic, can be defined for this decision, depending on the input set that the process provides. According to Definition 5.5 of Chapter 5, the direct input set of decision `A` is defined as follows: $I_A = \{o_B, o_C\}$ with $I_A \in dirs_A$ and $o_B \in O_B$ and $o_C \in O_C$. Following the same Definition 5.5, the input requirements set for decision A can be constructed to obtain the following: $dirs_A = (\{i1, i2\}, \{i1, o_C\}, \{o_B, i2\}, \{o_B, o_C\})$ with $o_B \in O_B$ and $o_C \in O_C$. With this, all possible decision services for decision A in Figure 10.1 can be constructed based on Definitions 5.5 and 5.6 from Chapter 5. Similarly, decision services for decisions B and C are obtained:

DS_A^1 is a tuple $(\{i1, i2\}, O_A)$
with $s_A^1 = \{i1, i2\} \in dirs_A$
DS_A^2 is a tuple $(\{i1, o_C\}, O_A)$
with $s_A^2 = \{i1, o_C\} \in dirs_A$ and $o_C \in O_C$
DS_A^3 is a tuple $(\{o_B, i2\}, O_A)$
with $s_A^3 = \{o_B, i2\} \in dirs_A$ and $o_B \in O_B$
DS_A^4 is a tuple $(\{o_B, o_C\}, O_A)$
with $s_A^4 = \{o_B, o_C\} \in dirs_A$ and $o_B \in O_B$ and $o_C \in O_C$
DS_B^1 is a tuple $(\{i1\}, O_B)$
with $s_B^1 = \{i1\} \in dirs_B$
DS_C^1 is a tuple $(\{i2\}, O_C)$
with $s_C^1 = \{i2\} \in dirs_C$

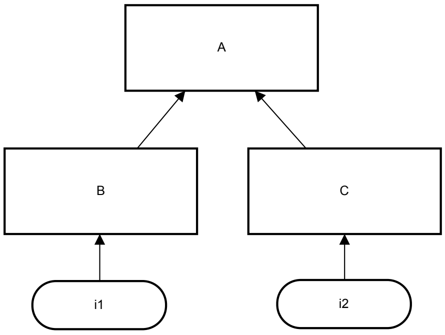


Figure 10.1: An example of a decision requirements diagram (DRD).

Note that due to the hierarchical graph structure of the DMN decision model, lower-level decision services can be composed into higher-level decision services. For instance, if DS_A^1 is invoked, lower-level decision services can be invoked by DS_A^1 in order to enact decision A. As such, DS_A^1 can call upon DS_B^1 and DS_C^1, since $s_B^1 \in s_A^1$ and $s_C^1 \in s_A^1$. This way, $o_B \in O_B$ and $o_C \in O_C$ are obtained, i.e., I_A, which can be used as inputs to enact decision A. This corresponds to invoking DS_A^4. As such, we have illustrated that DS_A^1 composes services DS_B^1 and DS_C^1 to provide inputs for the decision service DS_A^4, which is the equivalent of DS_A^1 due to the construction of the decision input requirements set for decision A

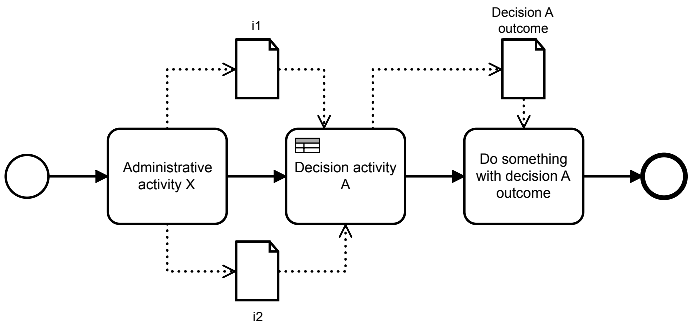

i1
Decision A outcome
Administrative activity X
Decision activity A
Do something with decision A outcome
i2

Figure 10.2: An example of a process using DS_A^1 of decision A from the decision model in Figure 10.1.

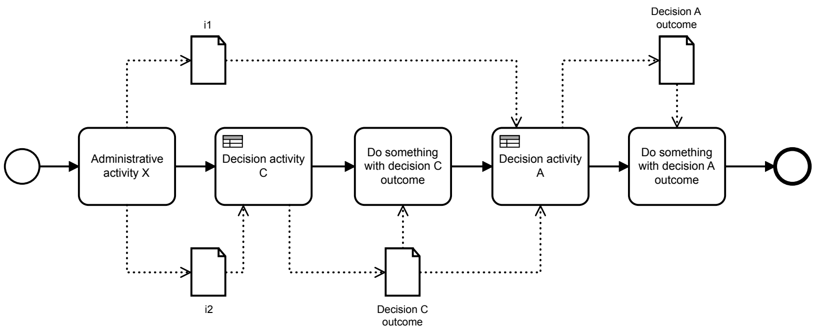

i1
Decision A outcome
Administrative activity X
Decision activity C
Do something with decision C outcome
Decision activity A
Do something with decision A outcome
i2
Decision C outcome

Figure 10.3: An example of a process using DS_A^2 of decision A from the decision model in Figure 10.1.

according to Definition 5.5 of Chapter 5. This composition allows for decision services to invoke lower-level decision services, thus avoiding the need for the replication of decision logic in equivalent decision services.

These different decision services provide processes with different ways of interacting with the underlying decision model. For instance, the process in Figure 10.2 uses DS_A^1 to interact with the decision model, while the process in Figure 10.3 uses DS_A^2. Similarly, the process in Figure 10.4 uses DS_A^3 and the process in Figure 10.5 uses DS_A^4. Note however, that other process variants are possible as well. For instance, in a process similar to the

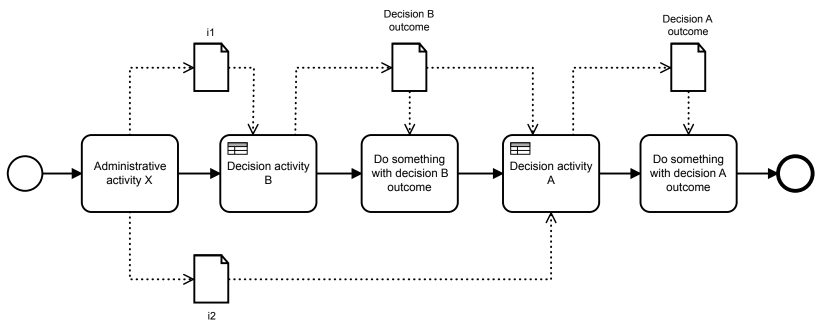

i1
Decision B outcome
Decision A outcome
Administrative activity X
Decision activity B
Do something with decision B outcome
Decision activity A
Do something with decision A outcome
i2

Figure 10.4: An example of a process using DS_A^3 of decision A from the decision model in Figure 10.1.

process in Figure 10.5, DS_A^4 may still be used without necessarily executing `Decision activity B` and `Decision activity C`, and consequently the decision services these decision activities pertain to, i.e., DS_B^1 and DS_C^1. In such a variant the outcomes of the lower-level decisions A and B are known without the need to invoke the subdecisions. Such a variant is given in Figure 10.6. Note that, according to Definition 5.10, this process variant only conforms to DS_A^4, while the variant given in Figure 10.5 conforms to all the decision services defined above for the decision model in Figure 10.1.

In analogy, the same can be done for the decision model in Figure 5.2: for each decision, possibly multiple decision services can be defined, thus allowing different process variants, or indeed entirely different processes and applications, to invoke the necessary decision as long as they provide an input set for that decision that is contained in the decision's input requirements set.

10.3 Part IV: Change Patterns, Model Evolution, and Tool Support

In this dissertation we base the decision model evolution on change patterns. Some of these change patterns are in part overlapping or equivalent. For instance, change patterns $\Delta\Pi 1$ -$\Delta\Pi 4$ in Table 7.1 of Chapter 7 handle the inclusion and exclusion of inputs and outputs

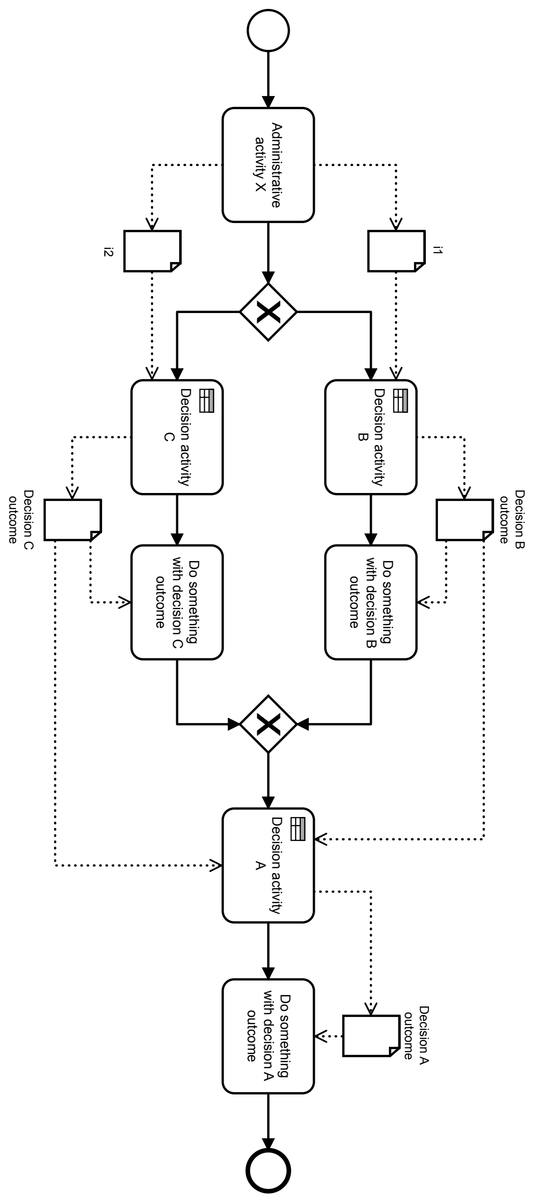

Administrative activity X
i1
i2
Decision activity B
Decision activity C
Decision B outcome
Decision C outcome
Do something with decision B outcome
Do something with decision C outcome
Decision activity A
Decision A outcome
Do something with decision A outcome

Figure 10.5: An example of a process using DS_A^4 of decision A from the decision model in Figure 10.1.

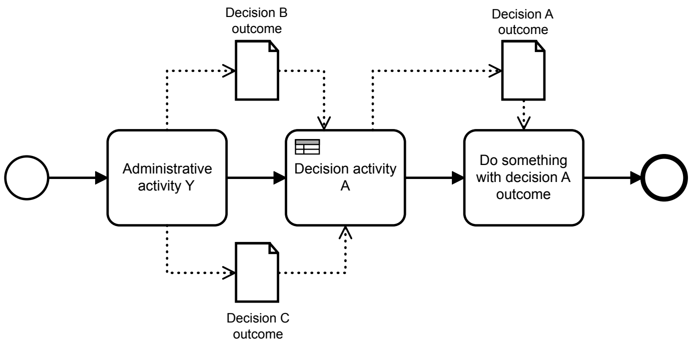


Figure 10.6: Another example of a process using DS_A^4 of decision A from the decision model in Figure 10.1.

in the decision table. Broadly speaking, these change patterns also influence the decision logic. As such, one can state that change pattern $\Delta\Pi 5$, i.e., the change in decision logic, partially overlaps with change patterns $\Delta\Pi 1$ -$\Delta\Pi 4$. However, we have defined the change in decision logic ($\Delta\Pi 5$) as a change in the mapping between the existing inputs and outputs. As such, the inclusion or exclusion of inputs and outputs is not considered a change in decision logic in the strict sense. We adopted this nuanced distinction of decision logic change in order to more precisely assess the effects of each change pattern. As is elaborated in Chapter 7, the change in decision logic in the strict sense ($\Delta\Pi 5$), i.e., changing the mapping between existing inputs and outputs, does not necessarily impact the consistency of the decision model, as long as the set of input values and the set of output values of a decision table remain the same. Applying one of the inclusion or exclusion change patterns ($\Delta\Pi 1$ -$\Delta\Pi 4$) on the other hand, most likely leads to induced inconsistencies in the referential integrity of the decision model. Therefore, this distinction of change patterns that leads to some overlap was needed to assess the effects of the respective change patterns on the decision hierarchy.

Similar overlaps in the defined change patterns were needed to assess the effects of administered changes on the different levels of the decision model, i.e., the decision requirements level and the decision logic level. For instance, some overlap exists with between, on the one hand, the input inclusion and exclusion patterns ($\Delta\Pi$*1* and $\Delta\Pi$*2*), and on the other hand, the input data node inclusion and exclusion patterns ($\Delta\Pi$*10* and $\Delta\Pi$*11*). The former refer to the decision logic level, while the latter pertain to the decision requirements level. Despite the overlap, both types of patterns need to be defined in order to assess their effects on all levels of the decision model and throughout the entire decision model hierarchy. Indeed, excluding an input data node from a decision requirements diagram leads to the exclusion of an input variable at the decision logic level, i.e., in a decision table. However, the opposite is not necessarily always true, as excluding an input variable from a decision table may also lead to the deletion of a subdecision that provides the input variable, and not to the deletion of an input data node. In order to safeguard such nuances and to enable the analysis of these changes in both levels of the decision model and throughout the decision model hierarchy, equivalent change patterns for both levels of the decision model need to be defined. Hence, despite a partial overlap between some change patterns, the patterns are not identical.

CHAPTER 11

Conclusions

"Sada je kraj pjesme
Sada je moje slovo cijelo"

("Now is the end of the song
Now my letter is whole")

Uspavanka
— *Mehmedalija Mak Dizdar*

This final chapter summarises this doctoral dissertation by reiterating the main contributions and limitations. Additionally, attention is given to future research opportunities that arise from the research presented. While specific contributions, limitations, and future research opportunities are discussed in the different chapters throughout this dissertation, this final chapter provides a brief and more general overview of these aspects.

11.1 Contributions

This dissertation revolves around the introduction of decisions as a systems modelling perspective in process-aware information systems in order to achieve decision-aware processes. By doing so, this dissertation acknowledges the importance of decision models in model-driven process systems engineering since organisations are increasingly relying on decision knowledge for the development, execution, evolution, and automation of their processes. Therefore, consistently integrating decisions and processes is of paramount importance for a sound development and management of knowledge-based processes. In this dissertation, the integration of decisions into processes is handled in a holistic fashion, meaning that decisions in processes are viewed as modular interconnected parts that can occur across the entire process and, as such, introduce long-distance decision dependencies into the process that need to be accounted for when aiming at a sound integration. While the contributions and implications of this dissertation are manifold, they can be categorised into three main areas, all of them advancing the overall contribution.

Firstly, principles and guidelines for the one-on-one integrated modelling of process and decision models are established and evaluated against cases from industry and literature, spanning different application areas. Decision models are viewed as given and static, and the process model is tailored to fit the decision model. As such, processes are augmented with decision intelligence which can influence the process from different perspectives, among others, the process control flow, process termination, process automation, as well as the data propagation throughout the process.

Secondly, a Service-Oriented Architecture (SOA) approach to decisions in processes is contributed, thus cementing decision services in a well-established theoretical framework. Here too, the decision model is viewed as given and static. However, the introduction of decision services specifies the different ways processes can interact with the decision model. Thus, rather than a one-on-one integration, here we consider the integration of a given and static decision model with processes in general. Additionally, the effective modelling of decisions in processes is linked to the automated discovery of data-aware processes, illustrating that decisions do manifest themselves as services in real-life enriched event logs.

Thirdly, decision model change patterns for dynamic system evolution are determined, allowing to evolve decision models while safeguarding their consistency. As such, unlike the in the two previous parts, here we consider dynamic decision models, i.e., models that are not static and that can change over time. Likewise, unlike in the first two parts, this part does not assume that the decision model is always correct, as change patterns can induce inconsistencies into the model. Moreover, the dependencies between decision model change patterns and decision-aware process models are unveiled and their effects on the consistency between the decision and process model are mitigated. All these aspects are supported by prototype modelling environments with built-in consistency verification capabilities, feedback mechanisms, and automated actions for consistency recovery.

In conclusion, this dissertation provides methods for a sound development of decision-aware processes along with mechanisms for the consistent evolution of decision models and decision-aware processes. Hence, this dissertation acknowledges the need for continuous decision and process evolution and improvement in knowledge-based systems.

11.2 Limitations

Introducing a decision perspective into process-aware information systems implies the introduction of an additional model, i.e., the decision model, into the system. This entails the introduction of additional complexity since new types of models are introduced. Additionally, the dependencies between the models in the system, i.e., the integration between process models and decision models, are a source of additional complexity in re-engineering processes for decision-awareness. This process re-engineering for decision-awareness implies eliminating the decision constructs that were present in the legacy process models. Additionally, the redesigned process models need to be compatible with the newly established decision models. Hence, consistency verification mechanisms need to be put in place to ensure a sound system integration.

Consequently it can be stated that, while the decision-aware approach to process management provides a plethora of advantages as elaborated upon in this dissertation, it introduces an additional cost in terms of higher system complexity. As such, a trade-off exists between on the one hand, introducing decision-awareness into processes by externalising decision constructs into separate decision models and subsequently redesigning the process to consistently integrate it with its newly established underlying decision model; and on the other hand, establishing additional forms of complexity by introducing new types of models and dependencies between different models. When a low level of decision-awareness is required, the systems engineer may opt to include the decision constructs into the process model itself. As such, no additional complexity is introduced into the system for merely a marginal gain in decision-awareness [71].

Additionally, it is worth emphasising that process re-engineering is necessary to achieve decision-aware processes. The transition from processes that convolute decision constructs with process constructs to processes that are aligned with an external decision model may prove to be a challenge in complex organisational settings. Rather than an abrupt transition, a step-wise and gradual transition between the two paradigms might be advisable in order to ensure a smooth process transformation.

Moreover, while this dissertation provides new methods for the development of decision-aware processes, additional empirical evaluation is indispensable for the support of the theoretical aspects contributed in this dissertation. These empirical evaluation aspects will be elucidated in the future research section which follows below.

11.3 Future Research Directions

Empirical evaluation on the understandability, maintainability, and complexity of decision-aware processes as introduced in this dissertation is required. The empirical information obtained for decision-aware processes should be compared to the empirical information obtained for regular processes that convolute process and decision constructs in a single model. As such, the advantages and disadvantages of decision-aware processes can be quantified and compared against regular processes.

The prototype modelling environments introduced in this dissertation provide additional opportunities for empirical evaluation. A comparison between modelling with and without the feedback mechanisms and automatic consistency recovery actions in the modelling environments can be performed. This way, it can be investigated whether the modelling support built into the modelling environments provides sufficient advantages over modelling freely without the feedback mechanisms enabled.

The feedback mechanisms and automatic consistency restore actions built into the modelling environments can be improved by relying on applied human-computer interaction research. More specifically, eye tracking studies can be performed to investigate the manner in which the modellers interact with the automated feedback mechanisms and how this influences their modelling capacity. Insights into redesigning the user-interface of the modelling environments or the way the feedback is provided to the modeller can be obtained.

An additional area for research is the development of a methodology for giving consistent guidelines for modelling DMN decision models, as well as integrated BPMN and DMN models. The DMN modelling methodology should provide guidelines and steps towards constructing a sound decision model. In particular, attention should be given to model granularity, i.e., the composition and decomposition of decisions within the decision model. In essence, one could model every DMN model as a single decision node in the decision requirements diagram. As such, that decision node encapsulates the decision logic entirely. Alternatively, one can opt to decompose this large decision table into smaller ones, effectively creating a hierarchy of decision tables, i.e., multiple decision nodes within the decision requirements diagram. Finding a balance between a single, entirely composed decision table, and a very granular, entirely decomposed decision model is a challenge that needs to be tackled. Note that in transitioning between these two situations complexity is moved from one level of the decision model to the other: in cases of a single decision table, the size and complexity of the decision requirements diagram is minimised, while the underlying decision table is capturing most of the complexity. In a more granular and decomposed decision model, the decision tables are smaller and simpler, while the decision requirements diagram boasts more elements and relations between these elements, thus increasing its complexity. A modelling methodology could for instance impose decomposition on parts of the logic that is often used standalone. As such, the modular design of the decision requirements diagram would facilitate the reusability of those standalone pieces of logic.

Similarly, a modelling methodology for the balance of DMN invocations within a BPMN model needs to be established. When considering a granular decision requirements diagram whose top-level decision is of interest to the BPMN process, a methodology should determine which, if any, of the subdecisions of the decision model should explicitly be invoked in the process model through decision activities. As such, a balance should be determined on the inclusion and exclusion of decision model links from the process model. In order to minimise the impact on process size and complexity only the necessary subdecisions should be included, i.e., the subdecisions whose standalone logic impacts one of the perspectives of the process. This impact can be on the resource perspective and handover of work if different decisions are carried out by different workers or systems or if they impact subsequent activities allocted to different resources. Similarly, subdecision invocations may need to be included in the process if they impact the control flow perspective of the process or if their intermediate results are used explicitly in a part of the process.

While this thesis handles the integration of process models and decision models, the combination of DMN with other requirements engineering approaches and other models may provide possibilities for additional research opportunities. Researchers have already combined knowledge representation at strategic, tactical and operational levels by integrating goal models, business use case models, and business process models [143, 144]. Since DMN is mainly used to model operational decisions it is situated at the operational level of knowledge representation. However, as this dissertation has argued, decisions are often the drivers of processes and they can impact the processes from multiple perspectives. Therefore, approaches that combine knowledge representation at strategic, tactical and operational levels can benefit by incorporating DMN into the mix. For that purpose the business use case model can be expanded to include an element that represents decisions at a tactical level. In analogy with tracing business use case realisations and business processes, by introducing a tactical decision element, tracing decisions at a tactical and operational level becomes a possibility. A further mapping between business use case elements and DMN can be established by for instance mapping actors to knowledge sources. As such, a knowledge worker or a system can for instance take a decision which drives a business use case, which on its turn supports a business goal, which at the end contributes to a business objective.

Furthermore, a remaining research gap is the integration of process models, decision models, and data models. Both established [54, 125] and recent [135] research on the integration of static data models and dynamic behavioural models exists. The inclusion of decision models into this integration is still lacking and can prove to be an interesting research direction in order to explicitly connect processes and decisions to data models, rather than merely including some data elements in the process model as is often the case.

This dissertation focused on the integration of decisions and procedural process models. However, declarative process models may prove to be a better fit for knowledge-intensive processes where flexibility in execution is required. Since knowledge-intensive processes often rely on decisions as well, the integration of declarative processes and decisions emerges as a promising research vain in decision-aware information systems. Furthermore, mixed-paradigm process modelling, which combines procedural and declarative process aspects into a single process model, provides processes that span a spectrum ranging from entirely structured to completely loosely coupled process models. Such mixed-paradigm models are capable of capturing a bigger array of realities. Integrating decisions with mixed-paradigm process models can consequently be achieved by consolidating the results provided in this dissertation with the results of the integration between decisions and declarative process models.

11.4 Final Word

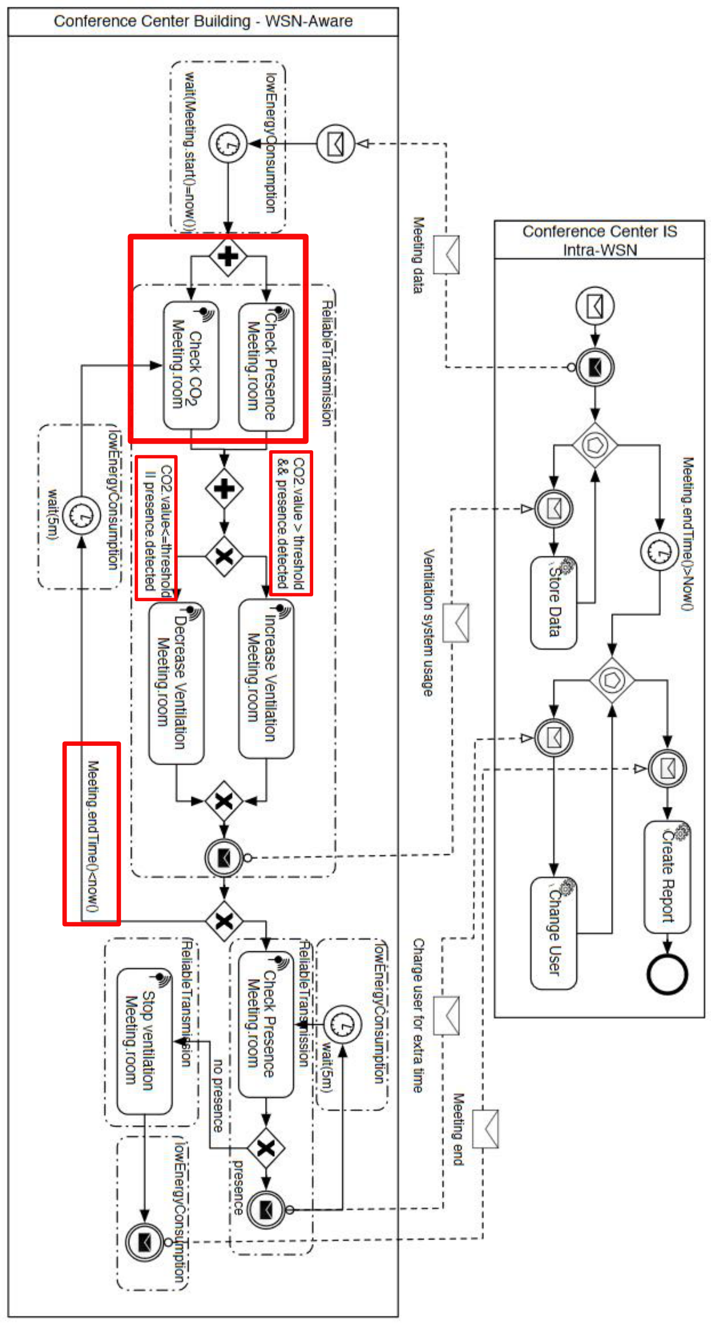
I hope that you enjoyed reading this dissertation as much as I have enjoyed writing it. Since research is an everlasting journey of improvement, I am sure that a lot of concepts in this book are subject to enhancement in the period to come. Therefore, I hope this work has inspired you to augment the horizon of this research or to embark on your own journey of discovery.

Faruk Hasić
May, 2020

APPENDIX A

Additional Models

This appendix provides additional models that are referred to from Chapter 4.



Conference Center Building - WSN-Aware
lowEnergyConsumption
wait(Meeting.start()=now())
Meeting data
ReliableTransmission
Check Presence Meeting.room
Check CO2 Meeting.room
CO2.value > threshold && presence.detected
CO2.value<=threshold || presence.detected
lowEnergyConsumption
wait(5m)
Increase Ventilation Meeting.room
Decrease Ventilation Meeting.room
Meeting.endTime()<now()
Ventilation system usage
lowEnergyConsumption
wait(5m)
ReliableTransmission
Check Presence Meeting.room
ReliableTransmission
Stop ventilation Meeting.room
no presence
presence
lowEnergyConsumption
Charge user for extra time
Meeting end
Conference Center IS Intra-WSN
Meeting.endTime()>Now()
Store Data
Change User
Create Report

Figure A.1: Original model from from [130] with errors indicated in red.

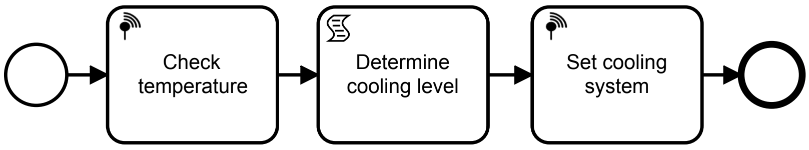


Figure A.2: Case 1 BPMN model with script task.

```
<scriptTask id="coolingLevel">
    <extensionElements>
        <inputOutput>
            <input temperature = -100..100>
                <output level = ['off', 'low', 'med', 'high', 'max']>
                    <script>
                        def level
                        switch (temperature) {
                            case -100..<0:
                                level = 'off'
                                break
                            case 0..<2:
                                level = 'low'
                                break
                            case 2..<5:
                                level = 'med'
                                break
                            case 5..<10:
                                level = 'high'
                                break
                            case 10..100:
                                level = 'max'
                                break
                            }
                            level
                    </script>
                </outputParameter>
            </inputParameter>
        </inputOutput>
    </extensionElements>
</scriptTask>
```

Figure A.3: Case 1 script task code.

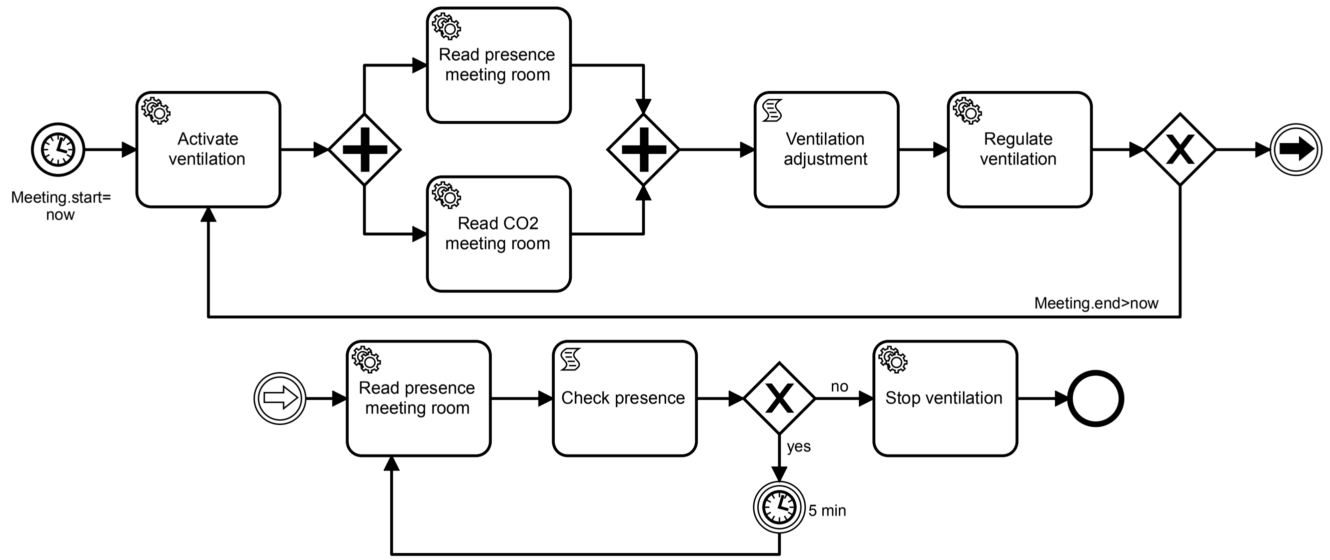
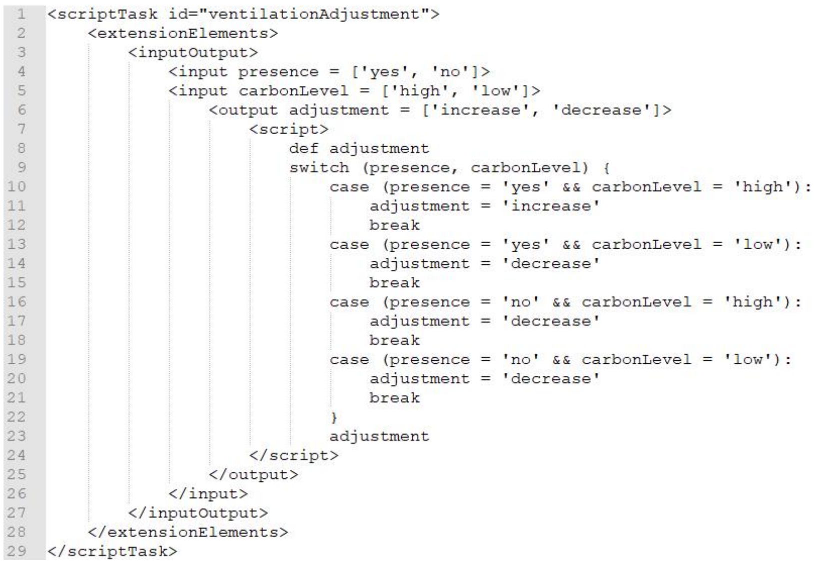
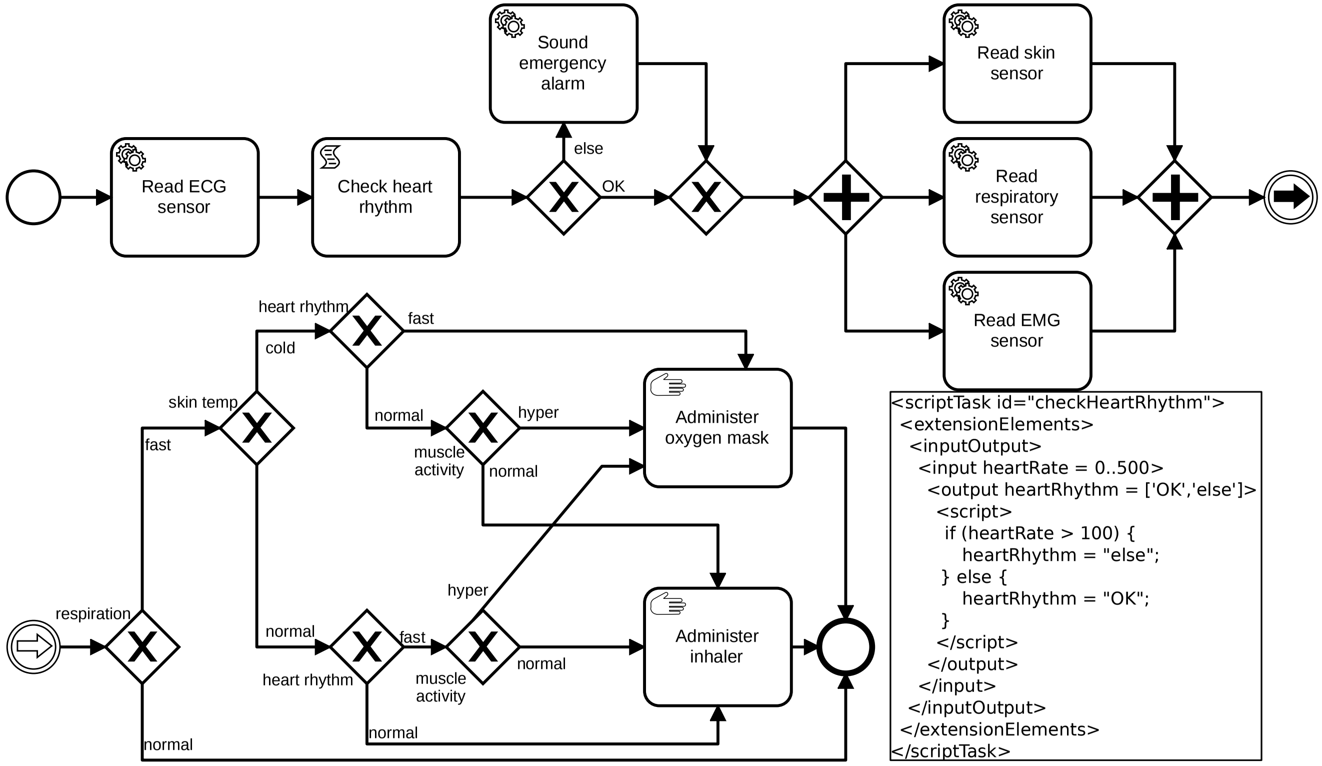


Figure A.4: Case 2 BPMN model with script task.



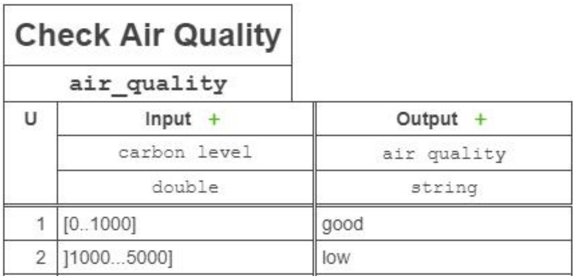
```
<scriptTask id="ventilationAdjustment">
    <extensionElements>
        <inputOutput>
            <input presence = ['yes', 'no']>
            <input carbonLevel = ['high', 'low']>
                <output adjustment = ['increase', 'decrease']>
                    <script>
                        def adjustment
                        switch (presence, carbonLevel) {
                            case (presence = 'yes' && carbonLevel = 'high'):
                                adjustment = 'increase'
                                break
                            case (presence = 'yes' && carbonLevel = 'low'):
                                adjustment = 'decrease'
                                break
                            case (presence = 'no' && carbonLevel = 'high'):
                                adjustment = 'decrease'
                                break
                            case (presence = 'no' && carbonLevel = 'low'):
                                adjustment = 'decrease'
                                break
                            }
                            adjustment
                    </script>
                </output>
            </input>
        </inputOutput>
    </extensionElements>
</scriptTask>
```

Figure A.5: Case 2 script task code.



```
<scriptTask id="checkHeartRhythm">
 <extensionElements>
  <inputOutput>
   <input heartRate = 0..500>
    <output heartRhythm = ['OK','else']>
     <script>
      if (heartRate > 100) {
        heartRhythm = "else";
      } else {
        heartRhythm = "OK";
      }
     </script>
    </output>
   </input>
  </inputOutput>
 </extensionElements>
</scriptTask>
```

Figure A.6: Case 2 BPMN model with script task and script task code.

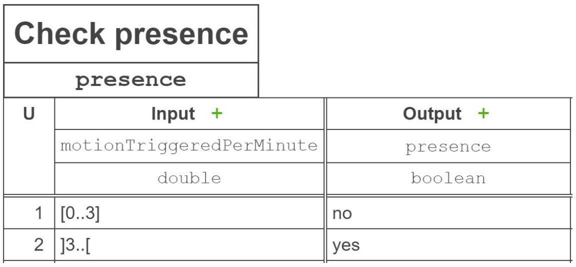


Check Air Quality

air_quality

U	Input +	Output +
	carbon level	air quality
	double	string
1	[0..1000]	good
2	]1000...5000]	low

Figure A.7: Case 2 air quality subdecision table.

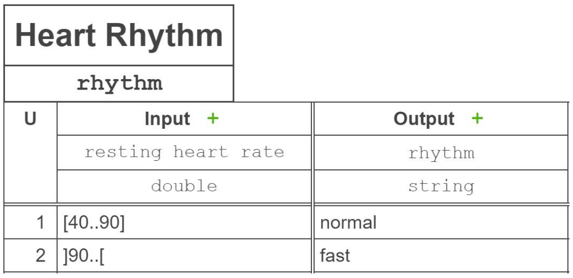


Check presence

presence

U	Input +	Output +
	motionTriggeredPerMinute	presence
	double	boolean
1	[0..3]	no
2	]3..[	yes

Figure A.8: Case 2 presence subdecision table.

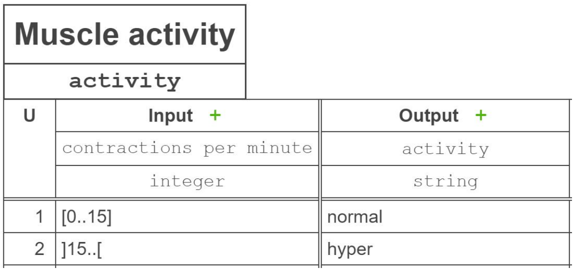


Heart Rhythm

rhythm

U	Input +	Output +
	resting heart rate	rhythm
	double	string
1	[40..90]	normal
2	]90..[	fast

Figure A.9: Case 3 heart rhythm subdecision table.

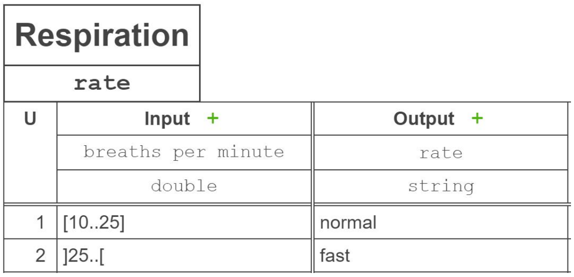


Muscle activity

activity

U	Input +	Output +
	contractions per minute	activity
	integer	string
1	[0..15]	normal
2	]15..[	hyper

Figure A.10: Case 3 muscle activity subdecision table.

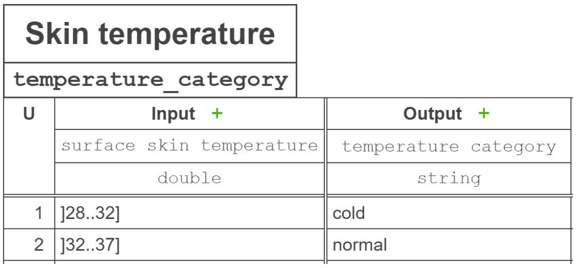


Respiration

rate

U	Input +	Output +
	breaths per minute	rate
	double	string
1	[10..25]	normal
2	]25..[	fast

Figure A.11: Case 3 respiration subdecision table.



Skin temperature

temperature_category

U	Input +	Output +
	surface skin temperature	temperature category
	double	string
1	]28..32]	cold
2	]32..37]	normal

Figure A.12: Case 3 skin temperature subdecision table.

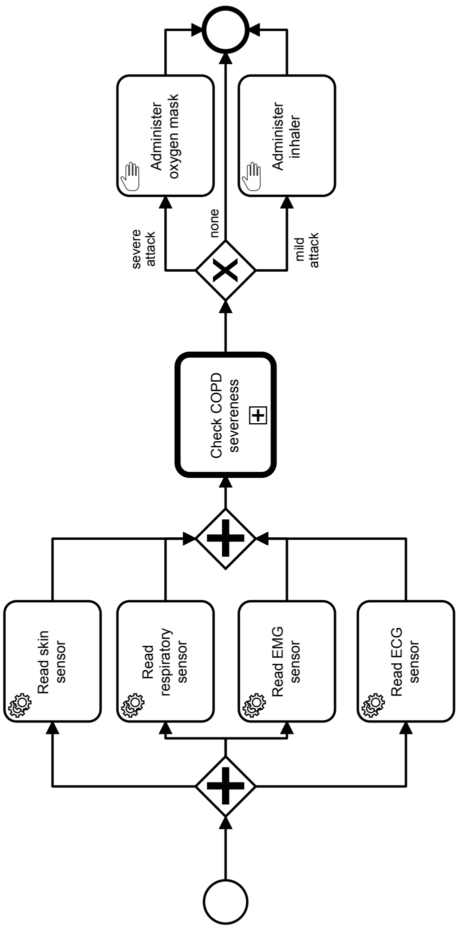


Figure A.13: Case 3 BPMN process model with call activity (subprocess).

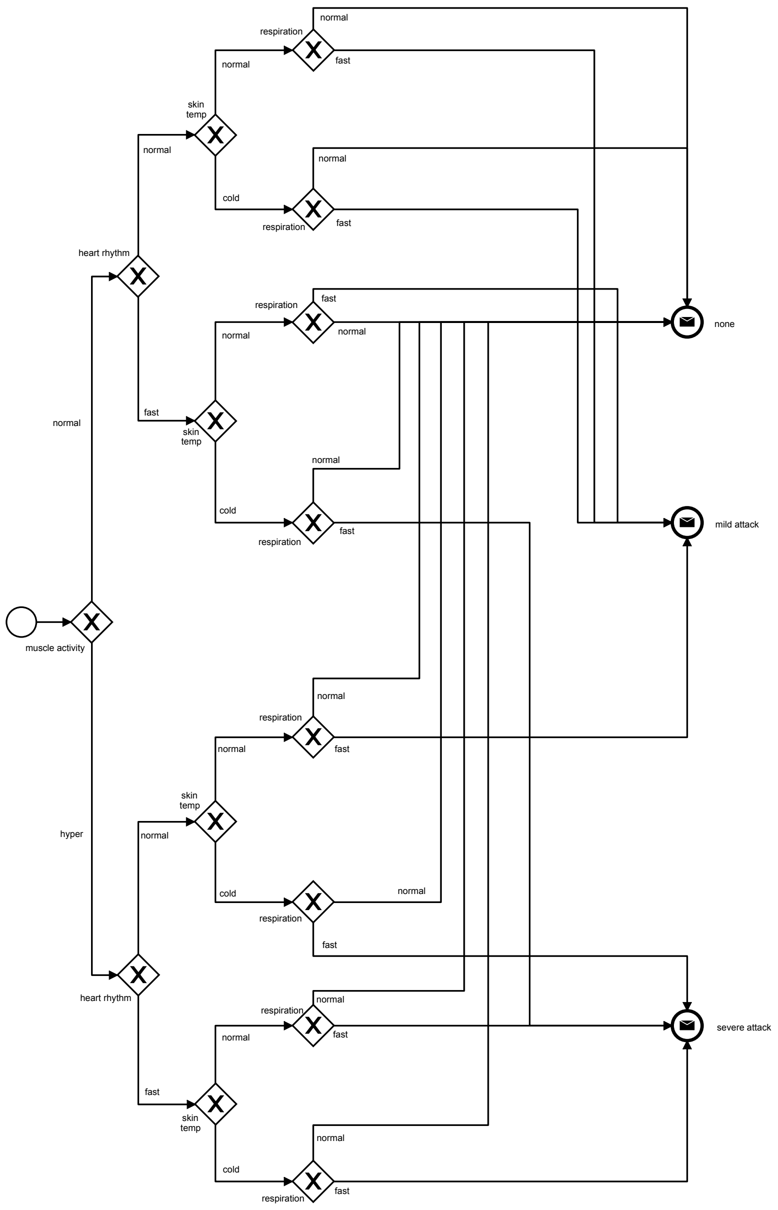


Figure A.14: Case 3 Check COPD Severeness callable BPMN decision subprocess.

APPENDIX B

Conforming Trace Cluster

This appendix provides a fragment of the mined trace cluster relating to the leftmost decision service in Figure 5.4 of Section 5.8. The full mined process models representing the trace clusters can be viewed online:
`https://feb.kuleuven.be/public/u0111379/TSC/`.

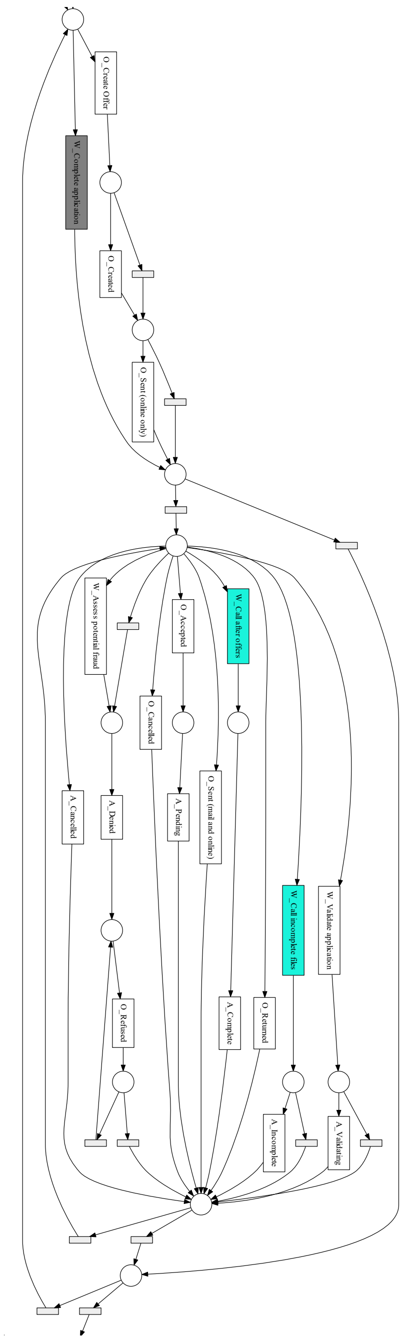


Figure B.1: Fragment of the trace cluster adhering to the decision service.

APPENDIX C

DMN Metamodel

This appendix gives an overview of the DMN metamodel, as specified in the latest DMN standard specification [106].

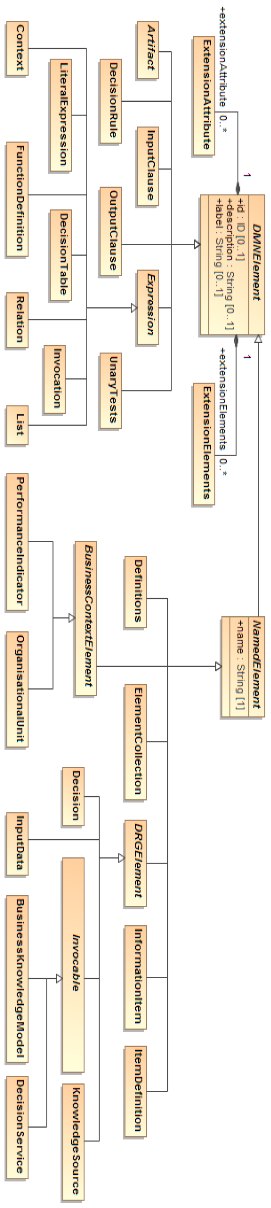


Figure C.1: Overview of the DMN meta model.

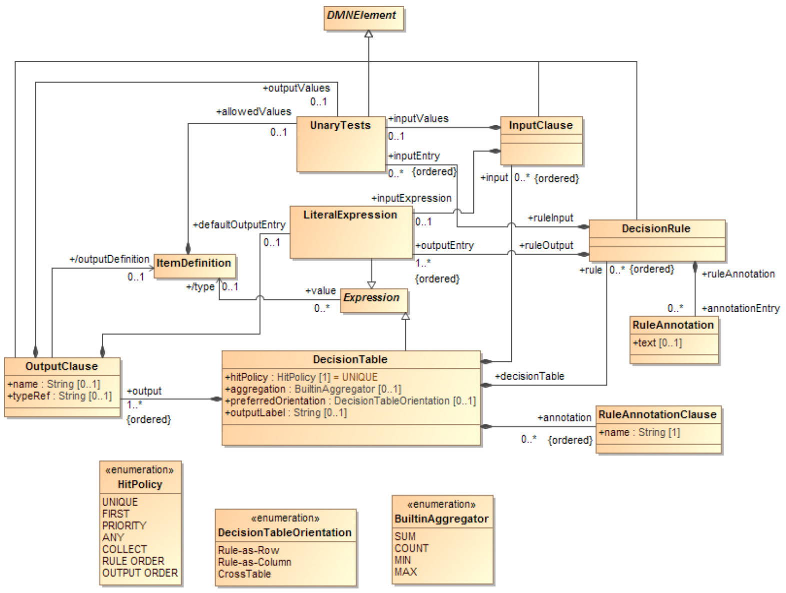


Figure C.2: Decision table meta model.

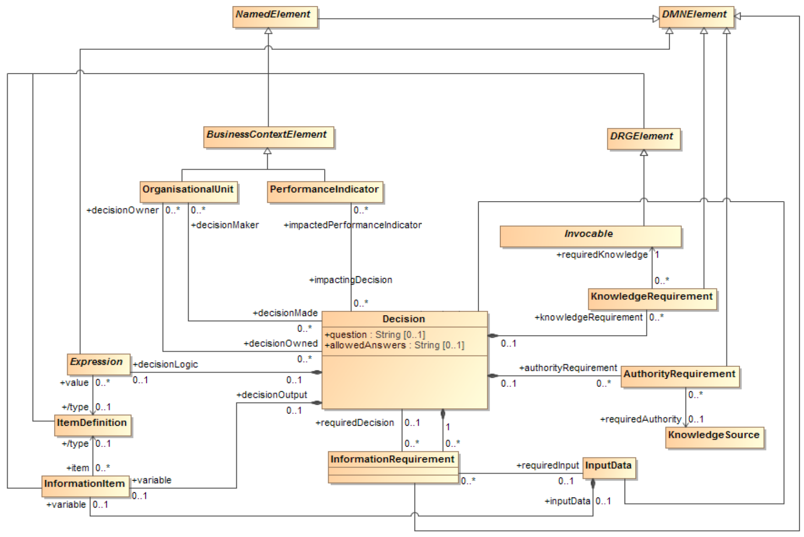


Figure C.3: Decision meta model.

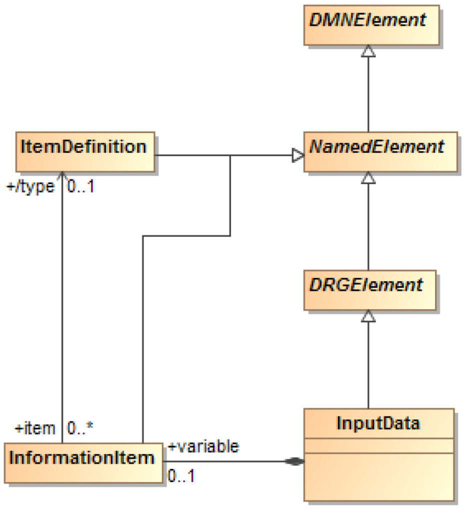


Figure C.4: Input data meta model.

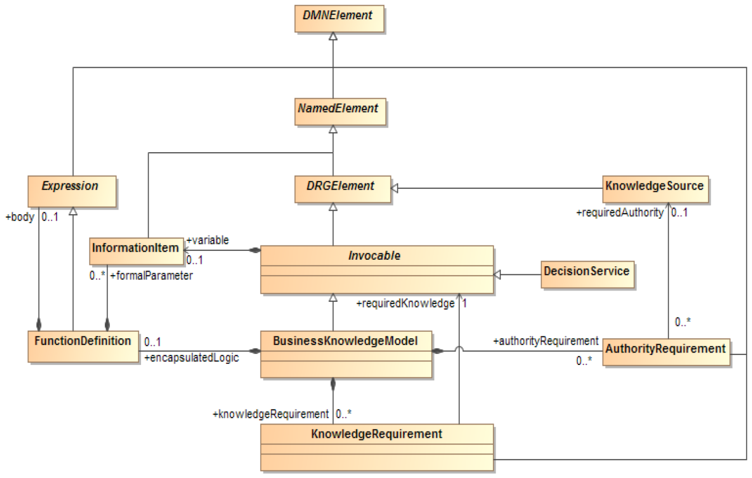


Figure C.5: Business knowledge model meta model.

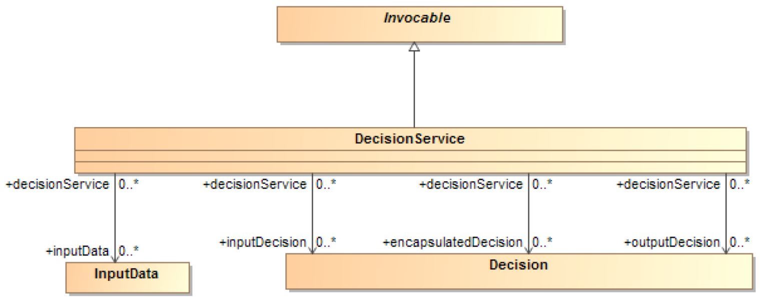


Figure C.6: Decision service meta model.

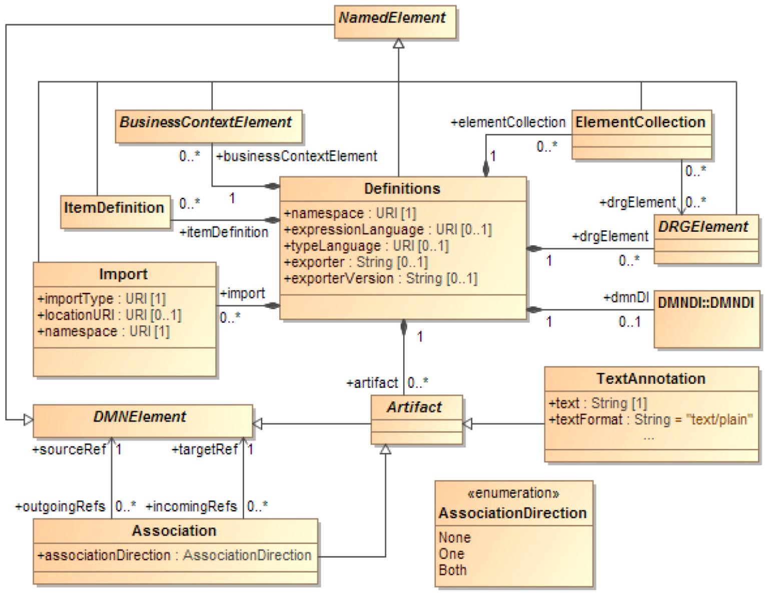


Figure C.7: Definitions meta model.

List of Figures

List of Tables

List of Algorithms

Bibliography

[1] Eram Abbasi and Kashif Abbasi. Business process modeling and decision model integration. In *2013 5th International Conference on Information and Communication Technologies*, pages 1–7, 2013. doi: 10.1109/ICICT.2013.6732789. URL `https://ieeexplore.ieee.org/abstract/document/6732789`.

[2] Clara Ayora, Victoria Torres, Barbara Weber, Manfred Reichert, and Vicente Pelechano. Enhancing modeling and change support for process families through change patterns. In Selmin Nurcan, Henderik Alex Proper, Pnina Soffer, John Krogstie, Rainer Schmidt, Terry A. Halpin, and Ilia Bider, editors, *Enterprise, Business-Process and Information Systems Modeling - 14th International Conference, BPMDS 2013, 18th International Conference, EMMSAD 2013, Held at CAiSE 2013, Valencia, Spain, June 17-18, 2013. Proceedings*, volume 147 of *Lecture Notes in Business Information Processing*, pages 246–260. Springer, 2013. doi: 10.1007/978-3-642-38484-4_18. URL `https://doi.org/10.1007/978-3-642-38484-4_18`.

[3] Clara Ayora, Victoria Torres, Barbara Weber, Manfred Reichert, and Vicente Pelechano. VIVACE: A framework for the systematic evaluation of variability support in process-aware information systems. *Information & Software Technology*, 57:248–276, 2015. doi: 10.1016/j.infsof.2014.05.009. URL `https://doi.org/10.1016/j.infsof.2014.05.009`.

[4] Clara Ayora, Victoria Torres, Jose Luis de la Vara, and Vicente Pelechano. Variability management in process families through change patterns. *Information & Software Technology*, 74:86–104, 2016. doi: 10.1016/j.infsof.2016.01.007. URL `https://doi.org/10.1016/j.infsof.2016.01.007`.

[5] Kimon Batoulis and Mathias Weske. A tool for checking soundness of decision-aware business processes. In Robert Clarisó, Henrik Leopold, Jan Mendling, Wil M. P. van der Aalst, Akhil Kumar, Brian T. Pentland, and Mathias Weske, editors, *Proceedings of the BPM Demo Track and BPM Dissertation Award co-located with 15th International Conference on Business Process Modeling (BPM 2017), Barcelona, Spain, September 13, 2017*, volume 1920 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2017. URL `http://ceur-ws.org/Vol-1920/BPM_2017_paper_184.pdf`.

[6] Kimon Batoulis and Mathias Weske. Soundness of decision-aware business processes. In Josep Carmona, Gregor Engels, and Akhil Kumar, editors, *Business Process Management Forum - BPM Forum 2017, Barcelona, Spain, September 10-15, 2017, Proceedings*, volume 297 of *Lecture Notes in Business Information Processing*, pages 106–124. Springer, 2017. doi: 10.1007/978-3-319-65015-9_7. URL `https://doi.org/10.1007/978-3-319-65015-9_7`.

[7] Kimon Batoulis and Mathias Weske. Disambiguation of DMN decision tables. In Witold Abramowicz and Adrian Paschke, editors, *Business Information Systems - 21st International Conference, BIS 2018, Berlin, Germany, July 18-20, 2018, Proceedings*, volume 320 of *Lecture Notes in Business Information Processing*, pages 236–249. Springer, 2018. doi: 10.1007/978-3-319-93931-5_17. URL `https://doi.org/10.1007/978-3-319-93931-5_17`.

[8] Kimon Batoulis and Mathias Weske. A tool for the uniqueification of DMN decision tables. In Wil M. P. van der Aalst, Fabio Casati, Raffaele Conforti, Massimiliano de Leoni, Marlon Dumas, Akhil Kumar, Jan Mendling, Surya Nepal, Brian T. Pentland, and Barbara Weber, editors, *Proceedings of the Dissertation Award, Demonstration, and Industrial Track at BPM 2018 co-located with 16th International Conference on Business Process Management (BPM 2018), Sydney, Australia, September 9-14, 2018*, volume 2196 of *CEUR Workshop Proceedings*, pages 116–119. CEUR-WS.org, 2018. URL `http://ceur-ws.org/Vol-2196/BPM_2018_paper_24.pdf`.

[9] Kimon Batoulis, Anne Baumgraß, Nico Herzberg, and Mathias Weske. Enabling dynamic decision making in business processes with DMN. In Manfred Reichert and Hajo A. Reijers, editors, *Business Process Management Workshops - BPM 2015, 13th International Workshops, Innsbruck, Austria, August 31 - September 3, 2015, Revised Papers*, volume 256 of *Lecture Notes*

in Business Information Processing, pages 418–431. Springer, 2015. doi: 10.1007/978-3-319-42887-1_34. URL `https://doi.org/10.1007/978-3-319-42887-1_34`.

[10] Kimon Batoulis, Stephan Haarmann, and Mathias Weske. Various notions of soundness for decision-aware business processes. In Heinrich C. Mayr, Giancarlo Guizzardi, Hui Ma, and Oscar Pastor, editors, *Conceptual Modeling - 36th International Conference, ER 2017, Valencia, Spain, November 6-9, 2017, Proceedings*, volume 10650 of *Lecture Notes in Computer Science*, pages 403–418. Springer, 2017. doi: 10.1007/978-3-319-69904-2_31. URL `https://doi.org/10.1007/978-3-319-69904-2_31`.

[11] Kimon Batoulis, Alexey Nesterenko, Guenther Repitsch, and Mathias Weske. Decision management in the insurance industry: Standards and tools. In Marco Brambilla and Thomas T. Hildebrandt, editors, *Proceedings of the BPM 2017 Industry Track co-located with the 15th International Conference on Business Process Management (BPM 2017), Barcelona, Spain, September 10-15, 2017*, volume 1985 of *CEUR Workshop Proceedings*, pages 52–63. CEUR-WS.org, 2017. URL `http://ceur-ws.org/Vol-1985/BPM17industry05.pdf`.

[12] Ekaterina Bazhenova, Susanne Bülow, and Mathias Weske. Discovering decision models from event logs. In Witold Abramowicz, Rainer Alt, and Bogdan Franczyk, editors, *Business Information Systems - 19th International Conference, BIS 2016, Leipzig, Germany, July, 6-8, 2016, Proceedings*, volume 255 of *Lecture Notes in Business Information Processing*, pages 237–251. Springer, 2016. doi: 10.1007/978-3-319-39426-8_19. URL `https://doi.org/10.1007/978-3-319-39426-8_19`.

[13] Ekaterina Bazhenova, Francesca Zerbato, Barbara Oliboni, and Mathias Weske. From BPMN process models to DMN decision models. *Information Systems*, 83:69–88, 2019. doi: 10.1016/j.is.2019.02.001. URL `https://doi.org/10.1016/j.is.2019.02.001`.

[14] Jörg Becker and Daniel Pfeiffer. Solving the conflicts of distributed process modelling: Towards an integrated approach. In Willie Golden, Thomas Acton, Kieran Conboy, Hans van der Heijden, and Virpi Kristiina Tuunainen, editors, *16th European Conference on Information Systems, ECIS 2008, Galway, Ireland, 2008*, pages 1555–1568, 2008. URL `http://aisel.aisnet.org/ecis2008/90`.

[15] Zohra Bellahsene. Schema evolution in data warehouses. *Knowledge and Information Systems*, 4(3):283–304, 2002. doi: 10.1007/s101150200008. URL `https://doi.org/10.1007/s101150200008`.

[16] Thierry Biard, Alexandre Le Mauff, Michel Bigand, and Jean Pierre Bourey. Separation of decision modeling from business process modeling using new "decision model and notation" (DMN) for automating operational decision-making. In Luis M. Camarinha-Matos, Frédérick Bénaben, and Willy Picard, editors, *Risks and Resilience of Collaborative Networks - 16th IFIP WG 5.5 Working Conference on Virtual Enterprises, PRO-VE 2015, Albi, France, October 5-7, 2015, Proceedings*, volume 463 of *IFIP Advances in Information and Communication Technology*, pages 489–496. Springer, 2015. doi: 10.1007/978-3-319-24141-8_45. URL `https://doi.org/10.1007/978-3-319-24141-8_45`.

[17] Thierry Biard, Jean Pierre Bourey, and Michel Bigand. DMN (decision model and notation) : De la modélisation à l'automatisation des décisions. In *Actes du XXXVème Congrès INFORSID, Toulouse, France, May 30 - June 2, 2017*, pages 327–342, 2017. URL `http://inforsid.fr/actes/2017/INFORSID_2017_paper_24.pdf`.

[18] Thierry Biard, Jean-Pierre Bourey, Michel Bigand, and Jean-Claude Bocquet. Modélisation des prises de décisions dans les processus métier grâce à DMN (Decision Model and Notation). In *12ème Congrès International de Génie Industriel 2017*, Compiègne, France, 2017. URL `https://hal.archives-ouvertes.fr/hal-01519433`.

[19] Guy Bieber and Jeff Carpenter. Introduction to service-oriented programming (rev 2.1). *OpenWings Whitepaper*, 2001. URL `http://lig-membres.imag.fr/donsez/ujf/m2r/glacs/services/ServiceOrientedIntroduction.pdf`.

[20] Gordon S. Blair, Nelly Bencomo, and Robert B. France. Models@run.time. *IEEE Computer*, 42(10):22–27, 2009. doi: 10.1109/MC.2009.326. URL `https://doi.org/10.1109/MC.2009.326`.

[21] Paolo Bocciarelli, Andrea D'Ambrogio, Andrea Giglio, and Emiliano Paglia. A BPMN extension for modeling cyber-physical-production-systems in the context of industry 4.0. In Giancarlo Fortino, MengChu Zhou, Zofia Lukszo, Athanasios V. Vasilakos, Francesco Basile, Carlos Enrique Palau, Antonio Liotta, Maria Pia

Fanti, Antonio Guerrieri, and Andrea Vinci, editors, *14th IEEE International Conference on Networking, Sensing and Control, ICNSC 2017, Calabria, Italy, May 16-18, 2017*, pages 599–604. IEEE, 2017. doi: 10.1109/ICNSC.2017.8000159. URL `https://doi.org/10.1109/ICNSC.2017.8000159`.

[22] Alexander Bock, Heiko Kattenstroth, and Sietse Overbeek. Towards a modeling method for supporting the management of organizational decision processes. In Hans-Georg Fill, Dimitris Karagiannis, and Ulrich Reimer, editors, *Modellierung 2014, 19.-21. März 2014, Wien, Österreich*, volume P-225 of *Lecture Notes in Informatics*, pages 49–64. GI, 2014. URL `https://dl.gi.de/20.500.12116/17065`.

[23] Barry W. Boehm, Bradford Clark, Ellis Horowitz, J. Christopher Westland, Raymond J. Madachy, and Richard W. Selby. Cost models for future software life cycle processes: COCOMO 2.0. *Annals of Software Engineering*, 1:57–94, 1995. doi: 10.1007/BF02249046. URL `https://doi.org/10.1007/BF02249046`.

[24] Dominik Bork, Dimitris Karagiannis, and Benedikt Pittl. Systematic analysis and evaluation of visual conceptual modeling language notations. In *12th International Conference on Research Challenges in Information Science, RCIS 2018, Nantes, France, May 29-31, 2018*, pages 1–11. IEEE, 2018. doi: 10.1109/RCIS.2018.8406652. URL `https://doi.org/10.1109/RCIS.2018.8406652`.

[25] Dominik Bork, Robert Buchmann, Dimitris Karagiannis, Moonkun Lee, and Elena-Teodora Miron. An open platform for modeling method conceptualization: The omilab digital ecosystem. *Communications of the Association for Information Systems*, 44: 673–697, 2019. ISSN 1529-3181. doi: 10.17705/1CAIS.04432. URL `http://eprints.cs.univie.ac.at/5462/`.

[26] Jerome Boyer and Hafedh Mili. *Agile Business Rule Development - Process, Architecture, and JRules Examples.* Springer, 2011. ISBN 978-3-642-19040-7. doi: 10.1007/978-3-642-19041-4. URL `https://doi.org/10.1007/978-3-642-19041-4`.

[27] Coral Calero, Mario Piattini, and Marcela Genero. Empirical validation of referential integrity metrics. *Information & Software Technology*, 43(15):949–957, 2001. doi: 10.1016/S0950-5849(01)00202-6. URL `https://doi.org/10.1016/S0950-5849(01)00202-6`.

[28] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Marco Montali, and Ario Santoso. Ontology-based governance of data-aware processes. In Markus Krötzsch and Umberto Straccia, editors, *Web Reasoning and Rule Systems - 6th International Conference, RR 2012, Vienna, Austria, September 10-12, 2012. Proceedings*, volume 7497 of *Lecture Notes in Computer Science*, pages 25–41. Springer, 2012. doi: 10.1007/978-3-642-33203-6_4. URL `https://doi.org/10.1007/978-3-642-33203-6_4`.

[29] Diego Calvanese, Marlon Dumas, Ülari Laurson, Fabrizio Maria Maggi, Marco Montali, and Irene Teinemaa. Semantics and analysis of DMN decision tables. In Marcello La Rosa, Peter Loos, and Oscar Pastor, editors, *Business Process Management - 14th International Conference, BPM 2016, Rio de Janeiro, Brazil, September 18-22, 2016. Proceedings*, volume 9850 of *Lecture Notes in Computer Science*, pages 217–233. Springer, 2016. doi: 10.1007/978-3-319-45348-4_13. URL `https://doi.org/10.1007/978-3-319-45348-4_13`.

[30] Diego Calvanese, Marlon Dumas, Fabrizio Maria Maggi, and Marco Montali. Semantic DMN: formalizing decision models with domain knowledge. In Stefania Costantini, Enrico Franconi, William Van Woensel, Roman Kontchakov, Fariba Sadri, and Dumitru Roman, editors, *Rules and Reasoning - International Joint Conference, RuleML+RR 2017, London, UK, July 12-15, 2017, Proceedings*, volume 10364 of *Lecture Notes in Computer Science*, pages 70–86. Springer, 2017. doi: 10.1007/978-3-319-61252-2_6. URL `https://doi.org/10.1007/978-3-319-61252-2_6`.

[31] Diego Calvanese, Marlon Dumas, Ülari Laurson, Fabrizio Maria Maggi, Marco Montali, and Irene Teinemaa. Semantics, analysis and simplification of DMN decision tables. *Information Systems*, 78:112–125, 2018. doi: 10.1016/j.is.2018.01.010. URL `https://doi.org/10.1016/j.is.2018.01.010`.

[32] Júlio Campos, Pedro H. Piccoli Richetti, Fernanda Araújo Baião, and Flávia Maria Santoro. Discovering business rules in knowledge-intensive processes through decision mining: An experimental study. In Ernest Teniente and Matthias Weidlich, editors, *Business Process Management Workshops - BPM 2017 International Workshops, Barcelona, Spain, September 10-11, 2017, Revised Papers*, volume 308 of *Lecture Notes in Business Information Processing*, pages 556–567. Springer, 2017. doi: 10.

1007/978-3-319-74030-0_44. URL https://doi.org/10.1007/978-3-319-74030-0_44.

[33] Bernardo Cánovas-Segura, Francesca Zerbato, Barbara Oliboni, Carlo Combi, Manuel Campos, Antonio Morales Nicolás, Jose M. Juarez, Roque Marín, and Francisco Palacios. A process-oriented approach for supporting clinical decisions for infection management. In *2017 IEEE International Conference on Healthcare Informatics, ICHI 2017, Park City, UT, USA, August 23-26, 2017*, pages 91–100. IEEE Computer Society, 2017. doi: 10.1109/ICHI.2017.73. URL https://doi.org/10.1109/ICHI.2017.73.

[34] Bernardo Cánovas-Segura, Francesca Zerbato, Barbara Oliboni, Carlo Combi, Manuel Campos, Antonio Morales Nicolás, Jose M. Juarez, Francisco Palacios, and Roque Marín. A decision support visualization tool for infection management based on BMPN and DMN. In Rafael Valencia-García, Katty Lagos-Ortiz, Gema Alcaraz-Mármol, Javier del Cioppo, Néstor Vera-Lucio, and Martha Bucaram-Leverone, editors, *Technologies and Innovation - Third International Conference, CITI 2017, Guayaquil, Ecuador, October 24-27, 2017, Proceedings*, volume 749 of *Communications in Computer and Information Science*, pages 158–168. Springer, 2017. doi: 10.1007/978-3-319-67283-0_12. URL https://doi.org/10.1007/978-3-319-67283-0_12.

[35] Ching-Yu Chen, Jui Hsi Fu, Today Sung, Ping-Feng Wang, Emery Jou, and Ming-Whei Feng. Complex event processing for the internet of things and its applications. In *2014 IEEE International Conference on Automation Science and Engineering, CASE 2014, New Taipei, Taiwan, August 18-22, 2014*, pages 1144–1149. IEEE, 2014. doi: 10.1109/CoASE.2014.6899470. URL https://doi.org/10.1109/CoASE.2014.6899470.

[36] Henry Chesbrough and Jim Spohrer. A research manifesto for services science. *Communications of the ACM*, 49(7):35–40, 2006. doi: 10.1145/1139922.1139945. URL https://doi.org/10.1145/1139922.1139945.

[37] Carlo Combi, Barbara Oliboni, Alessandro Zardiniy, and Francesca Zerbato. Seamless design of decision-intensive care pathways. In *2016 IEEE International Conference on Healthcare Informatics, ICHI 2016, Chicago, IL, USA, October 4-7, 2016*, pages 35–45. IEEE Computer Society, 2016. doi: 10.1109/ICHI.2016.9. URL https://doi.org/10.1109/ICHI.2016.9.

[38] Carlo Combi, Barbara Oliboni, Alessandro Zardini, and Francesca Zerbato. A methodological framework for the integrated design of decision-intensive care pathways—an application to the management of copd patients. *Journal of Healthcare Informatics Research*, 1(2):157–217, 2017. ISSN 2509-498X. doi: 10.1007/s41666-017-0007-4. URL `https://doi.org/10.1007/s41666-017-0007-4`.

[39] Carl Corea and Patrick Delfmann. A tool to monitor consistent decision-making in business process execution. In Wil M. P. van der Aalst, Fabio Casati, Raffaele Conforti, Massimiliano de Leoni, Marlon Dumas, Akhil Kumar, Jan Mendling, Surya Nepal, Brian T. Pentland, and Barbara Weber, editors, *Proceedings of the Dissertation Award, Demonstration, and Industrial Track at BPM 2018 co-located with 16th International Conference on Business Process Management (BPM 2018), Sydney, Australia, September 9-14, 2018*, volume 2196 of *CEUR Workshop Proceedings*, pages 76–80. CEUR-WS.org, 2018. URL `http://ceur-ws.org/Vol-2196/BPM_2018_paper_16.pdf`.

[40] Carl Corea, Jonas Blatt, and Patrick Delfmann. A tool for decision logic verification in DMN decision tables. In Benoît Depaire, Johannes De Smedt, Marlon Dumas, Dirk Fahland, Akhil Kumar, Henrik Leopold, Manfred Reichert, Stefanie Rinderle-Ma, Stefan Schulte, Stefan Seidel, and Wil M. P. van der Aalst, editors, *Proceedings of the Dissertation Award, Doctoral Consortium, and Demonstration Track at BPM 2019 co-located with 17th International Conference on Business Process Management, BPM 2019, Vienna, Austria, September 1-6, 2019*, volume 2420 of *CEUR Workshop Proceedings*, pages 169–173. CEUR-WS.org, 2019. URL `http://ceur-ws.org/Vol-2420/papeDT11.pdf`.

[41] Flavio Corradini, Alberto Polzonetti, and Oliviero Riganelli. Business rules in e-government applications. *CoRR*, abs/1802.08484, 2018. URL `http://arxiv.org/abs/1802.08484`.

[42] 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 - Second International Joint Conference, RuleML+RR 2018, Luxembourg, September 18-21, 2018, Proceedings*, volume 11092 of *Lecture Notes in Computer Science*, pages 248–263. Springer, 2018. doi: 10.1007/

978-3-319-99906-7_17. URL `https://doi.org/10.1007/978-3-319-99906-7_17`.

[43] Juliana Baptista dos Santos França, Joanne Manhães Netto, Juliana do E. Santo Carvalho, Flávia Maria Santoro, Fernanda Araujo Baião, and Mariano Gomes Pimentel. KIPO: the knowledge-intensive process ontology. *Software and Systems Modeling*, 14(3): 1127–1157, 2015. doi: 10.1007/s10270-014-0397-1. URL `https://doi.org/10.1007/s10270-014-0397-1`.

[44] Marlon Dumas, Marcello La Rosa, Jan Mendling, and Hajo A. Reijers. *Fundamentals of Business Process Management.* Springer, 2013. ISBN 978-3-642-33142-8. doi: 10.1007/978-3-642-33143-5. URL `https://doi.org/10.1007/978-3-642-33143-5`.

[45] Thomas Erl. *SOA: principles of service design.* Prentice Hall Press, 2007.

[46] Thomas Erl. *SOA design patterns.* Pearson Education, 2008.

[47] Ilker Etikan, Sulaiman Abubakar Musa, and Rukayya Sunusi Alkassim. Comparison of convenience sampling and purposive sampling. *American Journal of Theoretical and Applied Statistics*, 5(1):1–4, 2016. URL `http://ajtas.org/article?journalid=146&doi=10.11648/j.ajtas.20160501.11`.

[48] Dirk Fahland, Daniel Lübke, Jan Mendling, Hajo A. Reijers, Barbara Weber, Matthias Weidlich, and Stefan Zugal. Declarative versus imperative process modeling languages: The issue of understandability. In Terry A. Halpin, John Krogstie, Selmin Nurcan, Erik Proper, Rainer Schmidt, Pnina Soffer, and Roland Ukor, editors, *Enterprise, Business-Process and Information Systems Modeling, 10th International Workshop, BPMDS 2009, and 14th International Conference, EMMSAD 2009, held at CAiSE 2009, Amsterdam, The Netherlands, June 8-9, 2009. Proceedings*, volume 29 of *Lecture Notes in Business Information Processing*, pages 353–366. Springer, 2009. doi: 10.1007/978-3-642-01862-6_29. URL `https://doi.org/10.1007/978-3-642-01862-6_29`.

[49] Kathrin Figl, Jan Mendling, Gül Tokdemir, and Jan Vanthienen. What we know and what we do not know about DMN. *Enterprise Modelling and Information Systems Architectures*, 13:2:1–16, 2018. doi: 10.18417/emisa.13.2. URL `https://doi.org/10.18417/emisa.13.2`.

[50] Riadh Ghlala, Zahra Kodia Aouina, and Lamjed Ben Said. BPMN decision footprint: Towards decision harmony along BI process. In Giedre Dregvaite and Robertas Damasevicius, editors, *Information and Software Technologies - 22nd International Conference, ICIST 2016, Druskininkai, Lithuania, October 13-15, 2016, Proceedings*, volume 639 of *Communications in Computer and Information Science*, pages 269–284, 2016. doi: 10.1007/978-3-319-46254-7_22. URL `https://doi.org/10.1007/978-3-319-46254-7_22`.

[51] Stijn Goedertier and Jan Vanthienen. Compliant and flexible business processes with business rules. In Gil Regev, Pnina Soffer, and Rainer Schmidt, editors, *Proceedings of the CAISE*06 Workshop on Business Process Modelling, Development, and Support BPMDS '06, Luxemburg, June 5-9, 2006*, volume 236 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2006. URL `http://ceur-ws.org/Vol-236/paper3.pdf`.

[52] Jaap Gordijn, Hans Akkermans, and Hans van Vliet. Business modelling is not process modelling. In Stephen W. Liddle, Heinrich C. Mayr, and Bernhard Thalheim, editors, *Conceptual Modeling for E-Business and the Web, ER 2000 Workshops on Conceptual Modeling Approaches for E-Business and The World Wide Web and Conceptual Modeling, Salt Lake City, Utah, USA, October 9-12, 2000, Proceedings*, volume 1921 of *Lecture Notes in Computer Science*, pages 40–51. Springer, 2000. doi: 10.1007/3-540-45394-6_5. URL `https://doi.org/10.1007/3-540-45394-6_5`.

[53] Christian W. Günther and Wil M. P. van der Aalst. Fuzzy mining - adaptive process simplification based on multi-perspective metrics. In Gustavo Alonso, Peter Dadam, and Michael Rosemann, editors, *Business Process Management, 5th International Conference, BPM 2007, Brisbane, Australia, September 24-28, 2007, Proceedings*, volume 4714 of *Lecture Notes in Computer Science*, pages 328–343. Springer, 2007. doi: 10.1007/978-3-540-75183-0_24. URL `https://doi.org/10.1007/978-3-540-75183-0_24`.

[54] Raf Haesen, Monique Snoeck, Wilfried Lemahieu, and Stephan Poelmans. Existence dependency-based domain modeling for improving stateless process enactment. In *2009 IEEE Congress on Services, Part I, SERVICES I 2009, Los Angeles, CA, USA, July 6-10, 2009*, pages 515–521. IEEE Computer Society, 2009. doi: 10.1109/SERVICES-I.2009.19. URL `https://doi.org/10.1109/SERVICES-I.2009.19`.

[55] Alena Hallerbach, Thomas Bauer, and Manfred Reichert. *Configuration and Management of Process Variants*, pages 237–255. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. ISBN 978-3-642-00416-2. doi: 10.1007/978-3-642-00416-2_11. URL `https://doi.org/10.1007/978-3-642-00416-2_11`.

[56] Faruk Hasić and Estefanía Serral. Executing IoT processes in BPMN 2.0: Current support and remaining challenges. In Manuel Kolp, Jean Vanderdonckt, Monique Snoeck, and Yves Wautelet, editors, *13th International Conference on Research Challenges in Information Science, RCIS 2019, Brussels, Belgium, May 29-31, 2019*, pages 1–6. IEEE, 2019. doi: 10.1109/RCIS.2019.8876998. URL `https://doi.org/10.1109/RCIS.2019.8876998`.

[57] Faruk Hasić and Estefanía Serral. Change patterns for Decision Model and Notation (DMN) model evolution. In *The 18th Belgium-Netherlands Software Evolution Workshop (BENEVOL), Brussels, Belgium, 28-29 November, 2019*, pages 1–4. CEUR-WS.org, 2019.

[58] Faruk Hasić and Jan Vanthienen. Complexity metrics for DMN decision models. *Computer Standards & Interfaces*, 65:15–37, 2019. doi: 10.1016/j.csi.2019.01.006. URL `https://doi.org/10.1016/j.csi.2019.01.006`.

[59] 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*, 62(5): 2011–2028, 2020. ISSN 0219-3116. doi: 10.1007/s10115-019-01416-4. URL `https://doi.org/10.1007/s10115-019-01416-4`.

[60] Faruk Hasić, Johannes De Smedt, and Jan Vanthienen. An Illustration of Five Principles for Integrated Process and Decision Modelling (5PDM). Technical Report 1717, KU Leuven, 2017. URL `https://lirias.kuleuven.be/retrieve/466070`.

[61] Faruk Hasić, Lesly Devadder, Maxim Dochez, Jonas Hanot, Johannes De Smedt, and Jan Vanthienen. Challenges in refactoring processes to include decision modelling. In Ernest Teniente and Matthias Weidlich, editors, *Business Process Management Workshops - BPM 2017 International Workshops, Barcelona, Spain, September 10-11, 2017, Revised Papers*, volume 308 of *Lecture Notes in Business Information Processing*, pages 529–541. Springer, 2017. doi: 10.1007/978-3-319-74030-0\_42. URL `https://doi.org/10.1007/978-3-319-74030-0_42`.

[62] 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 - Confederated International Workshops, EI2N, FBM, ICSP, Meta4eS, OTMA 2017 and ODBASE Posters 2017, Rhodes, Greece, October 23-28, 2017, Revised Selected Papers*, volume 10697 of *Lecture Notes in Computer Science*, pages 259–269. Springer, 2017. doi: 10.1007/978-3-319-73805-5_28. URL `https://doi.org/10.1007/978-3-319-73805-5_28`.

[63] Faruk Hasić, Johannes De Smedt, and Jan Vanthienen. A service-oriented architecture design of decision-aware information systems: Decision as a service. In Hervé Panetto, Christophe Debruyne, Walid Gaaloul, Mike P. Papazoglou, Adrian Paschke, Claudio Agostino Ardagna, and Robert Meersman, editors, *On the Move to Meaningful Internet Systems. OTM 2017 Conferences - Confederated International Conferences: CoopIS, C&TC, and ODBASE 2017, Rhodes, Greece, October 23-27, 2017, Proceedings, Part I*, volume 10573 of *Lecture Notes in Computer Science*, pages 353–361. Springer, 2017. doi: 10.1007/978-3-319-69462-7_23. URL `https://doi.org/10.1007/978-3-319-69462-7_23`.

[64] Faruk Hasić, Johannes De Smedt, and Jan Vanthienen. Towards assessing the theoretical complexity of the decision model and notation (DMN). In Jens Gulden, Selmin Nurcan, Iris Reinhartz-Berger, Wided Guédria, Palash Bera, Sérgio Guerreiro, Michael Fellmann, and Matthias Weidlich, editors, *Joint Proceedings of the Radar tracks at the 18th International Working Conference on Business Process Modeling, Development and Support (BPMDS), and the 22nd International Working Conference on Evaluation and Modeling Methods for Systems Analysis and Development (EMMSAD), and the 8th International Workshop on Enterprise Modeling and Information Systems Architectures (EMISA) co-located with the 29th International Conference on Advanced Information Systems Engineering 2017 (CAiSE 2017), Essen, Germany, June 12-13, 2017*, volume 1859 of *CEUR Workshop Proceedings*, pages 64–71. CEUR-WS.org, 2017. URL `http://ceur-ws.org/Vol-1859/bpmds-07-paper.pdf`.

[65] Faruk Hasić, Linus Vanwijck, and Jan Vanthienen. Integrating processes, cases, and decisions for knowledge-intensive process

modelling. In Dominik Bork, Dimitris Karagiannis, and Jan Vanthienen, editors, *Proceedings of the 1st International Workshop on Practicing Open Enterprise Modeling within OMiLAB (PrOse 2017) co-located with 10th IFIP WG 8.1 Working Conference on the Practice of Enterprise Modelling (PoEM 2017), Leuven, Belgium, November 22, 2017*, volume 1999 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2017. URL http://ceur-ws.org/Vol-1999/paper2.pdf.

[66] Faruk Hasić, Johannes De Smedt, and Jan Vanthienen. Augmenting processes with decision intelligence: Principles for integrated modelling. *Decision Support Systems*, 107:1–12, 2018. doi: 10.1016/j.dss.2017.12.008. URL https://doi.org/10.1016/j.dss.2017.12.008.

[67] Faruk Hasić, Johannes De Smedt, and Jan Vanthienen. Redesigning processes for decision-awareness: Strategies for integrated modelling. In Antonia Bertolino, Vasco Amaral, Paulo Rupino, and Marco Vieira, editors, *11th International Conference on the Quality of Information and Communications Technology, QUATIC 2018, Coimbra, Portugal, September 4-7, 2018*, pages 247–250. IEEE Computer Society, 2018. doi: 10.1109/QUATIC.2018.00043. URL https://doi.org/10.1109/QUATIC.2018.00043.

[68] Faruk Hasić, Johannes De Smedt, Seppe vanden Broucke, and Estefanía Serral. A Parameter Assessment of Service-Oriented Architecture Process Mining Integrating Decisions (SOAP-MInD). Technical Report 1914, KU Leuven, 2019. URL https://lirias.kuleuven.be/retrieve/551259.

[69] Faruk Hasić, Carl Corea, Jonas Blatt, Patrick Delfmann, and Estefanía Serral. Decision model change patterns for dynamic system evolution. *Knowledge and Information Systems*, 2020. ISSN 0219-3116. doi: 10.1007/s10115-020-01469-w. URL https://doi.org/10.1007/s10115-020-01469-w.

[70] Faruk Hasić, Johannes De Smedt, Seppe Vanden Broucke, and Estefanía Serral. Decision as a Service (DaaS): A Service-Oriented Architecture Approach for Decisions in Processes. *IEEE Transactions on Services Computing*, pages 1–14, 2020. ISSN 2372-0204. URL https://doi.org/10.1109/TSC.2020.2965516.

[71] Faruk Hasić, Estefanía Serral, and Monique Snoeck. Comparing BPMN to BPMN + DMN for IoT Process Modelling: A Case-Based Inquiry. In *Proceeedings of the 35th ACM/SIGAPP Symposium On*

Applied Computing (SAC) 2020, Brno, Czech Republic, March 30-April 3, 2020, pages 53–60. Association for Computing Machinery (ACM), 2020.

[72] Moeen Hassanalieragh, Alex Page, Tolga Soyata, Gaurav Sharma, Mehmet Aktas, Gonzalo Mateos, Burak Kantarci, and Silvana Andreescu. Health monitoring and management using internet-of-things (iot) sensing with cloud-based processing: Opportunities and challenges. In *2015 IEEE International Conference on Services Computing, SCC 2015, New York City, NY, USA, June 27 - July 2, 2015*, pages 285–292. IEEE Computer Society, 2015. doi: 10.1109/SCC.2015.47. URL `https://doi.org/10.1109/SCC.2015.47`.

[73] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *MIS Quarterly*, 28 (1):75–105, 2004. URL `http://misq.org/design-science-in-information-systems-research.html`.

[74] Knut Hinkelmann. Business process flexibility and decision-aware modeling - the knowledge work designer. In Dimitris Karagiannis, Heinrich C. Mayr, and John Mylopoulos, editors, *Domain-Specific Conceptual Modeling, Concepts, Methods and Tools*, pages 397–414. Springer, 2016. doi: 10.1007/978-3-319-39417-6_18. URL `https://doi.org/10.1007/978-3-319-39417-6_18`.

[75] Knut Hinkelmann, Kyriakos Kritikos, Sabrina Kurjakovic, Benjamin Lammel, and Robert Woitsch. A modelling environment for business process as a service. In John Krogstie, Haralambos Mouratidis, and Jianwen Su, editors, *Advanced Information Systems Engineering Workshops - CAiSE 2016 International Workshops, Ljubljana, Slovenia, June 13-17, 2016, Proceedings*, volume 249 of *Lecture Notes in Business Information Processing*, pages 181–192. Springer, 2016. doi: 10.1007/978-3-319-39564-7_18. URL `https://doi.org/10.1007/978-3-319-39564-7_18`.

[76] Knut Hinkelmann, Arianna Pierfranceschi, and Emanuele Laurenzi. The knowledge work designer - modelling process logic and business logic. In Stefanie Betz and Ulrich Reimer, editors, *Modellierung 2016, 2.-4. März 2016, Karlsruhe - Workshopband*, volume P-255 of *Lecture Notes in Informatics*, pages 135–140. GI, 2016. URL `https://dl.gi.de/20.500.12116/846`.

[77] Flávio Eduardo Aoki Horita, Daniel Link, João Porto de Albuquerque, and Bernd Hellingrath. oDMN: An integrated model to

connect decision-making needs to emerging data sources in disaster management. In Tung X. Bui and Ralph H. Sprague Jr., editors, *49th Hawaii International Conference on System Sciences, HICSS 2016, Koloa, HI, USA, January 5-8, 2016*, pages 2882–2891. IEEE Computer Society, 2016. doi: 10.1109/HICSS.2016.361. URL `https://doi.org/10.1109/HICSS.2016.361`.

[78] Flávio Eduardo Aoki Horita, João Porto de Albuquerque, Victor Marchezini, and Eduardo M. Mendiondo. Bridging the gap between decision-making and emerging big data sources: An application of a model-based framework to disaster management in brazil. *Decision Support Systems*, 97:12–22, 2017. doi: 10.1016/j.dss.2017.03.001. URL `https://doi.org/10.1016/j.dss.2017.03.001`.

[79] Jing Hu, Ghazaleh Aghakhani, Faruk Hasić, and Estefanía Serral. An evaluation framework for design-time context-adaptation of process modelling languages. In Geert Poels, Frederik Gailly, Estefanía Serral, and Monique Snoeck, editors, *The Practice of Enterprise Modeling - 10th IFIP WG 8.1. Working Conference, PoEM 2017, Leuven, Belgium, November 22-24, 2017, Proceedings*, volume 305 of *Lecture Notes in Business Information Processing*, pages 112–125. Springer, 2017. doi: 10.1007/978-3-319-70241-4_8. URL `https://doi.org/10.1007/978-3-319-70241-4_8`.

[80] IBM. SOA fundamentals in a nutshell, 2008. URL `https://www.ibm.com/developerworks/webservices/tutorials/ws-soa-ibmcertified/ws-soa-ibmcertified.html`.

[81] IBM. PHP object orientation: Separating concerns, building more modular php applications. `https://www.ibm.com/developerworks/library/os-php-objectorient/os-php-objectorient-pdf.pdf`, 2009.

[82] IBM. New to SOA and web services, 2016. URL `https://www.ibm.com/developerworks/webservices/newto/index.html`.

[83] Christian Janiesch, Agnes Koschmider, Massimo Mecella, Barbara Weber, Andrea Burattin, Claudio Di Ciccio, Avigdor Gal, Udo Kannengiesser, Felix Mannhardt, Jan Mendling, Andreas Oberweis, Manfred Reichert, Stefanie Rinderle-Ma, WenZhan Song, Jianwen Su, Victoria Torres, Matthias Weidlich, Mathias Weske, and Liang Zhang. The internet-of-things meets business process management: Mutual benefits and challenges. *CoRR*, abs/1709.03628, 2017. URL `http://arxiv.org/abs/1709.03628`.

[84] Laurent Janssens, Ekaterina Bazhenova, Johannes De Smedt, Jan Vanthienen, and Marc Denecker. Consistent integration of decision (DMN) and process (BPMN) models. In Sergio España, Mirjana Ivanovic, and Milos Savic, editors, *Proceedings of the CAiSE'16 Forum, at the 28th International Conference on Advanced Information Systems Engineering (CAiSE 2016), Ljubljana, Slovenia, June 13-17, 2016*, volume 1612 of *CEUR Workshop Proceedings*, pages 121–128. CEUR-WS.org, 2016. URL `http://ceur-ws.org/Vol-1612/paper16.pdf`.

[85] Thomas Johanndeiter, Anat Goldstein, and Ulrich Frank. Towards business process models at runtime. In Nelly Bencomo, Robert B. France, Sebastian Götz, and Bernhard Rumpe, editors, *Proceedings of the 8th Workshop on Models @ Run.time co-located with 16th International Conference on Model Driven Engineering Languages and Systems (MODELS 2013), Miami, FL, USA, September 29, 2013*, volume 1079 of *CEUR Workshop Proceedings*, pages 13–25. CEUR-WS.org, 2013. URL `http://ceur-ws.org/Vol-1079/mrt13_submission_13.pdf`.

[86] Wan M.N. Wan Kadir and Pericles Loucopoulos. Linking and propagating business rule changes to is design. In *Information Systems Development*, pages 253–264. Springer, 2005. URL `https://link.springer.com/content/pdf/10.1007/0-387-28809-0_23.pdf`.

[87] William J. Kettinger and Varun Grover. Special section: Toward a theory of business process change management. *J. of Management Information Systems*, 12(1):9–30, 1995. URL `http://www.jmis-web.org/articles/534`.

[88] Krzysztof Kluza. Measuring complexity of business process models integrated with rules. In Leszek Rutkowski, Marcin Korytkowski, Rafal Scherer, Ryszard Tadeusiewicz, Lotfi A. Zadeh, and Jacek M. Zurada, editors, *Artificial Intelligence and Soft Computing - 14th International Conference, ICAISC 2015, Zakopane, Poland, June 14-28, 2015, Proceedings, Part II*, volume 9120 of *Lecture Notes in Computer Science*, pages 649–659. Springer, 2015. doi: 10.1007/978-3-319-19369-4_57. URL `https://doi.org/10.1007/978-3-319-19369-4_57`.

[89] Elena Kornyshova and Rébecca Deneckère. Decision-making ontology for information system engineering. In Jeffrey Parsons, Motoshi Saeki, Peretz Shoval, Carson C. Woo, and Yair Wand,

editors, *Conceptual Modeling - ER 2010, 29th International Conference on Conceptual Modeling, Vancouver, BC, Canada, November 1-4, 2010. Proceedings*, volume 6412 of *Lecture Notes in Computer Science*, pages 104–117. Springer, 2010. doi: 10.1007/978-3-642-16373-9_8. URL `https://doi.org/10.1007/978-3-642-16373-9_8`.

[90] Agnes Koschmider, Felix Mannhardt, and Tobias Heuser. On the contextualization of event-activity mappings. In Florian Daniel, Quan Z. Sheng, and Hamid Motahari, editors, *Business Process Management Workshops - BPM 2018 International Workshops, Sydney, NSW, Australia, September 9-14, 2018, Revised Papers*, volume 342 of *Lecture Notes in Business Information Processing*, pages 445–457. Springer, 2018. doi: 10.1007/978-3-030-11641-5_35. URL `https://doi.org/10.1007/978-3-030-11641-5_35`.

[91] Ülari Laurson and Fabrizio Maria Maggi. A tool for the analysis of DMN decision tables. In Leonardo Azevedo and Cristina Cabanillas, editors, *Proceedings of the BPM Demo Track 2016 Co-located with the 14th International Conference on Business Process Management (BPM 2016), Rio de Janeiro, Brazil, September 21, 2016*, volume 1789 of *CEUR Workshop Proceedings*, pages 56–60. CEUR-WS.org, 2016. URL `http://ceur-ws.org/Vol-1789/bpm-demo-2016-paper11.pdf`.

[92] Sihem Loukil, Slim Kallel, and Mohamed Jmaiel. An approach based on runtime models for developing dynamically adaptive systems. *Future Generation Comp. Syst.*, 68:365–375, 2017. doi: 10.1016/j.future.2016.07.006. URL `https://doi.org/10.1016/j.future.2016.07.006`.

[93] Felix Mannhardt, Massimiliano de Leoni, Hajo A. Reijers, and Wil M. P. van der Aalst. Data-driven process discovery - revealing conditional infrequent behavior from event logs. In Eric Dubois and Klaus Pohl, editors, *Advanced Information Systems Engineering*, pages 545–560. Springer International Publishing, 2017. ISBN 978-3-319-59536-8.

[94] Jan Mendling, Hajo A. Reijers, and Wil M. P. van der Aalst. Seven process modeling guidelines (7PMG). *Information & Software Technology*, 52(2):127–136, 2010. doi: 10.1016/j.infsof.2009.08.004. URL `https://doi.org/10.1016/j.infsof.2009.08.004`.

[95] Steven Mertens, Frederik Gailly, and Geert Poels. Enhancing declarative process models with DMN decision logic. In Khaled

Gaaloul, Rainer Schmidt, Selmin Nurcan, Sérgio Guerreiro, and Qin Ma, editors, *Enterprise, Business-Process and Information Systems Modeling - 16th International Conference, BPMDS 2015, 20th International Conference, EMMSAD 2015, Held at CAiSE 2015, Stockholm, Sweden, June 8-9, 2015, Proceedings*, volume 214 of *Lecture Notes in Business Information Processing*, pages 151–165. Springer, 2015. doi: 10.1007/978-3-319-19237-6_10. URL `https://doi.org/10.1007/978-3-319-19237-6_10`.

[96] Marinela Mircea, Bogdan Ghilic-Micu, and Marian Stoica. An agile architecture framework that leverages the strengths of business intelligence, decision management and service orientation. In *Business Intelligence-Solution for Business Development*, pages 15–32. InTech, 2012. URL `http://cdn.intechweb.org/pdfs/27300.pdf`.

[97] Angel Jiménez Molina, Jorge Gaete-Villegas, and Javier Fuentes. Profuso: Business process and ontology-based framework to develop ubiquitous computing support systems for chronic patients' management. *Journal of Biomedical Informatics*, 82:106–127, 2018. doi: 10.1016/j.jbi.2018.04.001. URL `https://doi.org/10.1016/j.jbi.2018.04.001`.

[98] Daniel L. Moody. The "physics" of notations: Toward a scientific basis for constructing visual notations in software engineering. *IEEE Transactions on Software Engineering*, 35(6):756–779, 2009. doi: 10.1109/TSE.2009.67. URL `https://doi.org/10.1109/TSE.2009.67`.

[99] Isel Moreno-Montes de Oca and Monique Snoeck. Pragmatic guidelines for business process modeling. *Available at SSRN 2592983*, pages 1–68, 2014. doi: 10.2139/ssrn.2592983. URL `https://ssrn.com/abstract=2592983`.

[100] Sabine Nagel, Carl Corea, and Patrick Delfmann. Effects of quantitative measures on understanding inconsistencies in business rules. In Tung Bui, editor, *52nd Hawaii International Conference on System Sciences, HICSS 2019, Grand Wailea, Maui, Hawaii, USA, January 8-11, 2019*, pages 1–10. ScholarSpace / AIS Electronic Library (AISeL), 2019. URL `http://hdl.handle.net/10125/59455`.

[101] Christoph Nagl, Florian Rosenberg, and Schahram Dustdar. VIDRE - A distributed service-oriented business rule engine based on ruleml. In *Tenth IEEE International Enterprise Distributed*

Object Computing Conference (EDOC 2006), 16-20 October 2006, Hong Kong, China, pages 35–44. IEEE Computer Society, 2006. doi: 10.1109/EDOC.2006.67. URL `https://doi.org/10.1109/EDOC.2006.67`.

[102] Natalya Fridman Noy and Michel C. A. Klein. Ontology evolution: Not the same as schema evolution. *Knowledge and Information Systems*, 6(4):428–440, 2004. URL `http://www.springerlink.com/index/10.1007/s10115-003-0137-2`.

[103] Bashar Nuseibeh, Jeff Kramer, and Anthony Finkelstein. A framework for expressing the relationships between multiple views in requirements specification. *IEEE Transactions on Software Engineering*, 20(10):760–773, 1994. doi: 10.1109/32.328995. URL `https://doi.org/10.1109/32.328995`.

[104] Lina Ochoa and Oscar González Rojas. Analysis and re-configuration of decision logic in adaptive and data-intensive processes. In Hervé Panetto, Christophe Debruyne, Walid Gaaloul, Mike P. Papazoglou, Adrian Paschke, Claudio Agostino Ardagna, and Robert Meersman, editors, *On the Move to Meaningful Internet Systems. OTM 2017 Conferences - Confederated International Conferences: CoopIS, C&TC, and ODBASE 2017, Rhodes, Greece, October 23-27, 2017, Proceedings, Part I*, volume 10573 of *Lecture Notes in Computer Science*, pages 306–313. Springer, 2017. doi: 10.1007/978-3-319-69462-7_20. URL `https://doi.org/10.1007/978-3-319-69462-7_20`.

[105] OMG. Business Process Model and Notation (BPMN). Version 2.0, Object Management Group, 2011. URL `https://www.omg.org/spec/BPMN/2.0/PDF/`.

[106] OMG. Decision Model and Notation (DMN). Version 1.2, Object Management Group, 2019. URL `https://www.omg.org/spec/DMN/1.2/PDF/`.

[107] Carlos Ordonez and Javier García-García. Referential integrity quality metrics. *Decision Support Systems*, 44(2):495–508, 2008. doi: 10.1016/j.dss.2007.06.004. URL `https://doi.org/10.1016/j.dss.2007.06.004`.

[108] José Miguel Pérez-Álvarez, María Teresa Gómez López, Luisa Parody, and Rafael M. Gasca. Process instance query language to include process performance indicators in DMN. In Remco M. Dijkman, Luís Ferreira Pires, and Stefanie Rinderle-Ma, editors,

20th IEEE International Enterprise Distributed Object Computing Workshop, EDOC Workshops 2016, Vienna, Austria, September 5-9, 2016, pages 1–8. IEEE Computer Society, 2016. doi: 10.1109/EDOCW.2016.7584381. URL `https://doi.org/10.1109/EDOCW.2016.7584381`.

[109] Lila Rao, Gunjan Mansingh, and Kweku-Muata Osei-Bryson. Building ontology based knowledge maps to assist business process re-engineering. *Decision Support Systems*, 52(3):577–589, 2012. doi: 10.1016/j.dss.2011.10.014. URL `https://doi.org/10.1016/j.dss.2011.10.014`.

[110] Christoph Rathfelder and Henning Groenda. isoamm: An independent SOA maturity model. In René Meier and Sotirios Terzis, editors, *Distributed Applications and Interoperable Systems, 8th IFIP WG 6.1 International Conference, DAIS 2008, Oslo, Norway, June 4-6, 2008. Proceedings*, volume 5053 of *Lecture Notes in Computer Science*, pages 1–15, 2008. doi: 10.1007/978-3-540-68642-2_1. URL `https://doi.org/10.1007/978-3-540-68642-2_1`.

[111] Manfred Reichert and Peter Dadam. Adept$_{\text{flex}}$-supporting dynamic changes of workflows without losing control. *Journal of Intelligent Information Systems*, 10(2):93–129, 1998. doi: 10.1023/A:1008604709862. URL `https://doi.org/10.1023/A:1008604709862`.

[112] Manfred Reichert and Peter Dadam. Enabling adaptive process-aware information systems with ADEPT2. In Jorge S. Cardoso and Wil M. P. van der Aalst, editors, *Handbook of Research on Business Process Modeling*, pages 173–203. IGI Global, 2009. doi: 10.4018/978-1-60566-288-6.ch008. URL `https://doi.org/10.4018/978-1-60566-288-6.ch008`.

[113] Manfred Reichert and Barbara Weber. *Ad hoc Changes of Process Instances*, pages 153–217. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-30409-5. doi: 10.1007/978-3-642-30409-5_7. URL `https://doi.org/10.1007/978-3-642-30409-5_7`.

[114] Stefanie Rinderle, Manfred Reichert, and Peter Dadam. Flexible support of team processes by adaptive workflow systems. *Distributed and Parallel Databases*, 16(1):91–116, 2004. doi: 10.1023/B:DAPD.0000026270.78463.77. URL `https://doi.org/10.1023/B:DAPD.0000026270.78463.77`.

[115] Willem De Roover and Jan Vanthienen. On the relation between decision structures, tables and processes. In Robert Meersman, Tharam S. Dillon, and Pilar Herrero, editors, *On the Move to Meaningful Internet Systems: OTM 2011 Workshops - Confederated International Workshops and Posters: EI2N+NSF ICE, ICSP+INBAST, ISDE, ORM, OTMA, SWWS+MONET+SeDeS, and VADER 2011, Hersonissos, Crete, Greece, October 17-21, 2011. Proceedings*, volume 7046 of *Lecture Notes in Computer Science*, pages 591–598. Springer, 2011. doi: 10.1007/978-3-642-25126-9_71. URL `https://doi.org/10.1007/978-3-642-25126-9_71`.

[116] Flávia Maria Santoro and Fernanda Araujo Baião. Knowledge-intensive process: A research framework. In Ernest Teniente and Matthias Weidlich, editors, *Business Process Management Workshops - BPM 2017 International Workshops, Barcelona, Spain, September 10-11, 2017, Revised Papers*, volume 308 of *Lecture Notes in Business Information Processing*, pages 460–468. Springer, 2017. doi: 10.1007/978-3-319-74030-0_36. URL `https://doi.org/10.1007/978-3-319-74030-0_36`.

[117] Flávia Maria Santoro, Tijs Slaats, Thomas T. Hildebrandt, and Fernanda Araújo Baião. Dcr-kipn a hybrid modeling approach for knowledge-intensive processes. In Alberto H. F. Laender, Barbara Pernici, Ee-Peng Lim, and José Palazzo M. de Oliveira, editors, *Conceptual Modeling - 38th International Conference, ER 2019, Salvador, Brazil, November 4-7, 2019, Proceedings*, volume 11788 of *Lecture Notes in Computer Science*, pages 153–161. Springer, 2019. doi: 10.1007/978-3-030-33223-5_13. URL `https://doi.org/10.1007/978-3-030-33223-5_13`.

[118] SAP. SAP decision service management, 2016. URL `https://www.sap.com/products/decision-service-management.html`.

[119] David Schumm, Frank Leymann, and Alexander Streule. Process viewing patterns. In *Proceedings of the 14th IEEE International Enterprise Distributed Object Computing Conference, EDOC 2010, Vitória, Brazil, 25-29 October 2010*, pages 89–98. IEEE Computer Society, 2010. doi: 10.1109/EDOC.2010.16. URL `https://doi.org/10.1109/EDOC.2010.16`.

[120] Estefanía Serral, Johannes De Smedt, Monique Snoeck, and Jan Vanthienen. Context-adaptive petri nets: Supporting adaptation for the execution context. *Expert Systems with Applications*, 42

(23):9307–9317, 2015. doi: 10.1016/j.eswa.2015.08.004. URL https://doi.org/10.1016/j.eswa.2015.08.004.

[121] Estefanía Serral, Pedro Valderas, and Vicente Pelechano. Addressing the evolution of automated user behaviour patterns by runtime model interpretation. *Software and Systems Modeling*, 14(4):1387–1420, 2015. doi: 10.1007/s10270-013-0371-3. URL https://doi.org/10.1007/s10270-013-0371-3.

[122] Johannes De Smedt, Faruk Hasić, Seppe K. L. M. vanden Broucke, and Jan Vanthienen. Towards a holistic discovery of decisions in process-aware information systems. In Josep Carmona, Gregor Engels, and Akhil Kumar, editors, *Business Process Management - 15th International Conference, BPM 2017, Barcelona, Spain, September 10-15, 2017, Proceedings*, volume 10445 of *Lecture Notes in Computer Science*, pages 183–199. Springer, 2017. doi: 10.1007/978-3-319-65000-5_11. URL https://doi.org/10.1007/978-3-319-65000-5_11.

[123] 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*, 183:1–15, 2019. doi: 10.1016/j.knosys.2019.104866. URL https://doi.org/10.1016/j.knosys.2019.104866.

[124] Koen Smit, Martijn Zoet, and Matthijs Berkhout. Verification capabilities for business rules management in the dutch governmental context. In *2017 International Conference on Research and Innovation in Information Systems (ICRIIS)*, pages 1–6, 2017. doi: 10.1109/ICRIIS.2017.8002499. URL https://ieeexplore.ieee.org/abstract/document/8002499/.

[125] Monique Snoeck, Stephan Poelmans, and Guido Dedene. An architecture for bridging OO and business process modeling. In *TOOLS 2000: 33rd International Conference on Technology of Object-Oriented Languages and Systems, 5-8 June 2000, St. Malo, France*, page 132. IEEE Computer Society, 2000. doi: 10.1109/TOOLS.2000.848757. URL https://doi.org/10.1109/TOOLS.2000.848757.

[126] Wei Song and Hans-Arno Jacobsen. Static and dynamic process change. *IEEE Transactions on Services Computing*, 11(1):215–231, 2018. doi: 10.1109/TSC.2016.2536025. URL https://doi.org/10.1109/TSC.2016.2536025.

[127] Sherry X. Sun, J. Leon Zhao, Jay F. Nunamaker Jr., and Olivia R. Liu Sheng. Formulating the data-flow perspective for business process management. *Inf. Syst. Res.*, 17(4):374–391, 2006. doi: 10.1287/isre.1060.0105. URL `https://doi.org/10.1287/isre.1060.0105`.

[128] Michael Szvetits and Uwe Zdun. Systematic literature review of the objectives, techniques, kinds, and architectures of models at runtime. *Software and Systems Modeling*, 15(1):31–69, 2016. doi: 10.1007/s10270-013-0394-9. URL `https://doi.org/10.1007/s10270-013-0394-9`.

[129] Feng Tian. *An information System for Food Safety Monitoring in Supply Chains based on HACCP, Blockchain and Internet of Things.* PhD thesis, WU Vienna University of Economics and Business, 2018. URL `https://epub.wu.ac.at/6090/`.

[130] Stefano Tranquillini, Patrik Spieß, Florian Daniel, Stamatis Karnouskos, Fabio Casati, Nina Oertel, Luca Mottola, Felix Jonathan Oppermann, Gian Pietro Picco, Kay Römer, and Thiemo Voigt. Process-based design and integration of wireless sensor network applications. In Alistair P. Barros, Avigdor Gal, and Ekkart Kindler, editors, *Business Process Management - 10th International Conference, BPM 2012, Tallinn, Estonia, September 3-6, 2012. Proceedings*, volume 7481 of *Lecture Notes in Computer Science*, pages 134–149. Springer, 2012. doi: 10.1007/978-3-642-32885-5_10. URL `https://doi.org/10.1007/978-3-642-32885-5_10`.

[131] Mohammed Hadi Valipour, Bavar Amirzafari, Khashayar Niki Maleki, and Negin Daneshpour. A brief survey of software architecture concepts and service oriented architecture. In *2009 2nd IEEE International Conference on Computer Science and Information Technology*, pages 34–38, 2009. doi: 10.1109/ICCSIT.2009.5235004. URL `https://ieeexplore.ieee.org/abstract/document/5235004`.

[132] Han van der Aa, Henrik Leopold, Kimon Batoulis, Mathias Weske, and Hajo A. Reijers. Integrated process and decision modeling for data-driven processes. In Manfred Reichert and Hajo A. Reijers, editors, *Business Process Management Workshops - BPM 2015, 13th International Workshops, Innsbruck, Austria, August 31 - September 3, 2015, Revised Papers*, volume 256 of *Lecture Notes in Business Information Processing*, pages 405–417. Springer, 2015.

doi: 10.1007/978-3-319-42887-1_33. URL https://doi.org/10.1007/978-3-319-42887-1_33.

[133] Wil M. P. van der Aalst. Service mining: Using process mining to discover, check, and improve service behavior. *IEEE Transactions on Services Computing*, 6(4):525–535, 2013. doi: 10.1109/TSC.2012.25. URL https://doi.org/10.1109/TSC.2012.25.

[134] Wil M. P. van der Aalst and Ernesto Damiani. Processes meet big data: Connecting data science with process science. *IEEE Transactions on Services Computing*, 8(6):810–819, 2015. doi: 10.1109/TSC.2015.2493732. URL https://doi.org/10.1109/TSC.2015.2493732.

[135] Wil M. P. van der Aalst, Guangming Li, and Marco Montali. Object-centric behavioral constraints, 2017. URL https://arxiv.org/pdf/1703.05740.pdf.

[136] Irene T. P. Vanderfeesten, Hajo A. Reijers, and Wil M. P. van der Aalst. Product based workflow support: Dynamic workflow execution. In Zohra Bellahsene and Michel Léonard, editors, *Advanced Information Systems Engineering, 20th International Conference, CAiSE 2008, Montpellier, France, June 16-20, 2008, Proceedings*, volume 5074 of *Lecture Notes in Computer Science*, pages 571–574. Springer, 2008. doi: 10.1007/978-3-540-69534-9_42. URL https://doi.org/10.1007/978-3-540-69534-9_42.

[137] Jan Vanthienen, Filip Caron, and Johannes De Smedt. Business rules, decisions and processes: five reflections upon living apart together. In *SIGBPS Workshop on Business Processes and Services*, pages 76–81, 2013. URL https://core.ac.uk/download/pdf/34582439.pdf.

[138] Barbara Von Halle and Larry Goldberg. *The decision model: a business logic framework linking business and technology.* Taylor & Francis, 2009.

[139] Barbara von Halle and Larry Goldberg. The decision model # 2: Improving process models and the requirements process. Technical report, Sapiens, 2010. URL https://www.sapiens.com/wp-content/uploads/2019/11/The-Decision-Model-2-Improving-Process-Models-and-the-Requirements-Process.pdf.

[140] Wei Wang, Marta Indulska, and Shazia Sadiq. Factors affecting business process and business rule integration. In *25th Australasian Conference on Information Systems, Auckland, New Zealand, 8-10 December, 2014*, pages 1–10, 2014. URL `https://openrepository.aut.ac.nz/bitstream/handle/10292/8060/acis20140_submission_44.pdf?sequence=1&isAllowed=y`.

[141] Wei Wang, Marta Indulska, and Shazia Sadiq. Guidelines for business rule modeling decisions. *Journal of Computer Information Systems*, 58(4):363–373, 2018. doi: 10.1080/08874417.2017.1285683. URL `https://doi.org/10.1080/08874417.2017.1285683`.

[142] Yi Wang, Jian Yang, Weiliang Zhao, and Jianwen Su. Change impact analysis in service-based business processes. *Service Oriented Computing and Applications*, 6(2):131–149, 2012. doi: 10.1007/s11761-011-0093-8. URL `https://doi.org/10.1007/s11761-011-0093-8`.

[143] Yves Wautelet and Stephan Poelmans. Aligning the elements of the RUP/UML business use-case model and the BPMN business process diagram. In Paul Grünbacher and Anna Perini, editors, *Requirements Engineering: Foundation for Software Quality - 23rd International Working Conference, REFSQ 2017, Essen, Germany, February 27 - March 2, 2017, Proceedings*, volume 10153 of *Lecture Notes in Computer Science*, pages 22–30. Springer, 2017. doi: 10.1007/978-3-319-54045-0_2. URL `https://doi.org/10.1007/978-3-319-54045-0_2`.

[144] Yves Wautelet and Stephan Poelmans. An integrated enterprise modeling framework using the RUP/UML business use-case model and BPMN. In Geert Poels, Frederik Gailly, Estefanía Serral Asensio, and Monique Snoeck, editors, *The Practice of Enterprise Modeling - 10th IFIP WG 8.1. Working Conference, PoEM 2017, Leuven, Belgium, November 22-24, 2017, Proceedings*, volume 305 of *Lecture Notes in Business Information Processing*, pages 299–315. Springer, 2017. doi: 10.1007/978-3-319-70241-4_20. URL `https://doi.org/10.1007/978-3-319-70241-4_20`.

[145] Barbara Weber, Stefanie Rinderle, and Manfred Reichert. Change patterns and change support features in process-aware information systems. In John Krogstie, Andreas L. Opdahl, and Guttorm Sindre, editors, *Advanced Information Systems Engineering, 19th International Conference, CAiSE 2007, Trondheim, Norway, June*

11-15, 2007, Proceedings, volume 4495 of *Lecture Notes in Computer Science*, pages 574–588. Springer, 2007. doi: 10.1007/978-3-540-72988-4_40. URL `https://doi.org/10.1007/978-3-540-72988-4_40`.

[146] Barbara Weber, Manfred Reichert, Jan Mendling, and Hajo A. Reijers. Refactoring large process model repositories. *Computers in Industry*, 62(5):467–486, 2011. doi: 10.1016/j.compind.2010.12.012. URL `https://doi.org/10.1016/j.compind.2010.12.012`.

[147] Geert Wets, Jan Vanthienen, and Harry J. P. Timmermans. Modelling decision tables from data. In Xindong Wu, Kotagiri Ramamohanarao, and Kevin B. Korb, editors, *Research and Development in Knowledge Discovery and Data Mining, Second Pacific-Asia Conference, PAKDD-98, Melbourne, Australia, April 15-17, 1998, Proceedings*, volume 1394 of *Lecture Notes in Computer Science*, pages 412–413. Springer, 1998. doi: 10.1007/3-540-64383-4_48. URL `https://doi.org/10.1007/3-540-64383-4_48`.

[148] Andreas Wombacher and Maarten Rozie. Evaluation of workflow similarity measures in service discovery. In Mareike Schoop, Christian Huemer, Michael Rebstock, and Martin Bichler, editors, *Service Oriented Electronic Commerce: Proceedings zur Konferenz im Rahmen der Multikonferenz Wirtschaftsinformatik, 20.-22. Februar 2006 in Passau, Deutschland*, volume P-80 of *Lecture Notes in Informatics*, pages 57–71. GI, 2006. URL `https://dl.gi.de/20.500.12116/24294`.

[149] Alaaeddine Yousfi, Marcin Hewelt, Christine Bauer, and Mathias Weske. Toward uBPMN-based patterns for modeling ubiquitous business processes. *IEEE Transactions on Industrial Informatics*, 14(8):3358–3367, 2018. doi: 10.1109/TII.2017.2777847. URL `https://doi.org/10.1109/TII.2017.2777847`.

[150] Alaaeddine Yousfi, Kimon Batoulis, and Mathias Weske. Achieving business process improvement via ubiquitous decision-aware business processes. *ACM Transactions on Internet Technology*, 19(1):14:1–14:19, 2019. doi: 10.1145/3298986. URL `https://doi.org/10.1145/3298986`.

[151] Alireza Zarghami, Brahmananda Sapkota, Mohammad Zarifi Eslami, and Marten van Sinderen. Decision as a service: Separating decision-making from application process logic. In Chi-Hung Chi, Dragan Gasevic, and Willem-Jan van den Heuvel, editors,

16th IEEE International Enterprise Distributed Object Computing Conference, EDOC 2012, Beijing, China, September 10-14, 2012, pages 103–112. IEEE Computer Society, 2012. doi: 10.1109/EDOC.2012.21. URL `https://doi.org/10.1109/EDOC.2012.21`.

Publication List

Articles in internationally reviewed scientific journals

1. **Faruk Hasić**, Carl Corea, Jonas Blatt, Patrick Delfmann, Estefanía Serral Asensio. Decision Model Change Patterns for Dynamic System Evolution. *Knowledge And Information Systems*, *article in press*, 2020.

2. **Faruk Hasić**, Johannes De Smedt, Seppe vanden Broucke, Estefanía Serral Asensio. Decision as a Service (DaaS): A Service-Oriented Architecture Approach for Decisions in Processes. *IEEE Transactions On Services Computing*, *article in press*, 2020.

3. **Faruk Hasić**, Jan Vanthienen. From Decision Knowledge to E-Government Expert Systems: The Case of Income Taxation for Foreign Artists in Belgium. *Knowledge And Information Systems*, 62 (5), 2011–2028, 2020.

4. Letha J. Sooter, Stephen Hasley, Robert Lario, Kenneth S. Rubin, **Faruk Hasić**. Modeling a Clinical Pathway for Contraception. *Applied Clinical Informatics*, 10 (5), 935-943, 2019.

5. **Faruk Hasić**, Jan Vanthienen. Complexity Metrics for DMN Decision Models. *Computer Standards & Interfaces*, 65, 15-37, 2019.

6. Johannes De Smedt, **Faruk Hasić**, Seppe vanden Broucke, Jan Vanthienen. Holistic Discovery of Decision Models from Process Execution Data. *Knowledge-Based Systems*, 183, 104866, 2019.

7. **Faruk Hasić**, Johannes De Smedt, Jan Vanthienen. Augmenting Processes with Decision Intelligence: Principles for Integrated Modelling. *Decision Support Systems*, 107, 1-12, 2018.

Papers at international conferences and symposia, published in full in proceedings

8. **Faruk Hasić**, Carl Corea, Jonas Blatt, Patrick Delfmann, Estefanía Serral Asensio. A Tool for the Verification of Decision Model and Notation (DMN) Models. *14th International Conference on Research Challenges in Information Science (RCIS), Demo Session*, Limassol (Cyprus), 2020.

9. Estefanía Serral Asensio, Caroline Vander Stede, **Faruk Hasić**. Leveraging IoT in Retail Industry: A Maturity Model, *IEEE International Conference on Business Informatics (IEEE CBI)*, Antwerp (Belgium), 2020.

10. **Faruk Hasić**, Monique Snoeck, Estefanía Serral Asensio. Comparing BPMN to BPMN + DMN for IoT Process Modelling: A Case-Based Inquiry. *35th ACM/SIGAPP Symposium on Applied Computing (SAC)*, Brno (Czech Republic) *article in press*, 2020.

11. **Faruk Hasić**, Estefanía Serral Asensio. Change Patterns for Decision Model and Notation (DMN) Model Evolution. *The 18th Belgium-Netherlands Software Evolution Workshop (BENEVOL)*, Brussels (Belgium), *article in press*, 2020.

12. **Faruk Hasić**, Estefanía Serral Asensio. Executing IoT Processes in BPMN 2.0: Current Support and Remaining Challenges. *13th IEEE International Conference on Research Challenges in Information Science (RCIS)*, Brussels (Belgium), 1-6, 2019.

13. **Faruk Hasić**, Alexander De Craemer, Thijs Hegge, Gideon Magala, Jan Vanthienen. Measuring the Complexity of DMN Decision Models. *Business Process Management Workshops at BPM 2018*, Sydney (Australia), 514-526, 2019.

14. Marjolein Deryck, **Faruk Hasić**, Jan Vanthienen. A Case-Based Inquiry into the Decision Model and Notation (DMN) and the Knowledge Base (KB) Paradigm. *2nd International Joint Conference on Rules and Reasoning (RULEML + RR)*, Luxembourg (Luxembourg), 248-263, 2018.

15. **Faruk Hasić**, Johannes De Smedt, Jan Vanthienen. Redesigning Processes for Decision-Awareness: Strategies for Integrated Modelling. *11th International Conference on the Quality of Information and Communications Technology (QUATIC)*, Coimbra (Portugal), 247-250, 2018.

16. **Faruk Hasić**, Johannes De Smedt, Jan Vanthienen. Developing a Modelling and Mining Framework for Integrated Processes and Decisions. *On the Move to Meaningful Internet Systems - OTM Academy - OTM 2017 Workshops*, Rhodes (Greece), 259-269, 2018.

17. **Faruk Hasić**, Lesly Devadder, Maxim Dochez, Jonas Hanot, Johannes De Smedt, Jan Vanthienen. Challenges in Refactoring Processes to Include Decision Modelling. *Business Process Management Workshops at BPM 2017*, Barcelona (Spain), 529-541, 2018.

18. **Faruk Hasić**, Johannes De Smedt, Jan Vanthienen. A Service-Oriented Architecture Design of Decision-Aware Information Systems: Decision as a Service. *25th International Conference on Cooperative Information Systems (CoopIS)*, Rhodes (Greece), 353-361, 2017.

19. Jing Hu, Ghazaleh Aghakhani, **Faruk Hasić**, Estefanía Serral Asensio. An Evaluation Framework for Design-Time Context-Adaptation of Process Modelling Languages. *10th Conference on the Practice of Enterprise Modelling (PoEM)*, Leuven (Belgium), 112-125, 2017.

20. **Faruk Hasić**, Linus Vanwijck, Jan Vanthienen. Integrating Processes, Cases, and Decisions for Knowledge-Intensive Process Modelling. *1st International Workshop on Practicing Open Enterprise Modelling (PrOse 2017) @PoEM*, Leuven (Belgium), 13-24, 2017.

21. Johannes De Smedt, **Faruk Hasić**, Seppe vanden Broucke, Jan Vanthienen. Towards a Holistic Discovery of Decisions in Process-Aware Information Systems. *15th International Conference on Business Process Management (BPM)*, Barcelona (Spain), 183-199, 2017.

22. **Faruk Hasić**, Johannes De Smedt, Jan Vanthienen. Towards Assessing the Theoretical Complexity of the Decision Model and Notation (DMN). *International Working Conference on Business Process Modeling, Development and Support (BPMDS) @CAiSE*, Essen (Germany), 64-71, 2017.

Technical reports published by the author's home institution

23. **Faruk Hasić**, Johannes De Smedt, Seppe vanden Broucke, Estefanía Serral Asensio. A Parameter Assessment of Service-Oriented Architecture Process Mining Integrating Decisions (SOAP-MInD). *FEB Research Report KBI_1914 (KU Leuven)*, Leuven (Belgium), 1-9, 2019.

24. **Faruk Hasić**, Johannes De Smedt, Jan Vanthienen. An illustration of Five Principles for Integrated Process and Decision Modelling (5PDM). *FEB Research Report KBI_1717 (KU Leuven)*, Leuven (Belgium), 1-8, 2017.

Meeting abstracts and posters presented at international scientific conferences and symposia

25. Jan Vanthienen, Thibaut Bender, **Faruk Hasić**. The Support of Decision Modeling Features and Concepts in Tooling. *Decision Camp*, Luxembourg (Luxembourg), 2018.

26. **Faruk Hasić**. Towards Integrated Process-Decision Modelling and Mining. *On the Move to Meaningful Internet Systems - OTM Academy*, Rhodes (Greece), 2017.

Articles submitted for publication in internationally reviewed scientific journals

27. Tevye Jacobs, Estefanía Serral Asensio, **Faruk Hasić**. A Reference Architecture for IoT-Enhanced Business Processes. *Computer Standards & Interfaces*, *submitted*, 2020.

Working papers to be submitted for publication

28. **Faruk Hasić**, Carl Corea, Jonas Blatt, Patrick Delfmann, Estefanía Serral Asensio. Consistent Evolution of Integrated Process and Decision Models. *To be submitted for publication*, 2020.

29. **Faruk Hasić**, Angel Jiménez-Molina, Estefanía Serral Asensio. A Process Analytics Approach for Multi-Perspective Understanding of Large Healthcare Processes with an Application in Clinical Radiology. *To be submitted for publication*, 2020.

30. Carl Corea, Jonas Blatt, **Faruk Hasić**, Estefanía Serral Asensio, Patrick Delfmann. A Tool for Consistency Verification Between Process (BPMN) and Decision (DMN) Models. *To be submitted for publication*, 2020.

Author Biography

Faruk HASIĆ was born on June 19, 1993 in Srebrenica, Bosnia and Herzegovina. He obtained a B.Sc. in Business Engineering from Ghent University in 2014, a B.Sc. in Business and Information Systems Engineering from KU Leuven in 2015, and an M.Sc. in Business Engineering from KU Leuven in 2016, with a major in quantitative methods and a minor in informatics. In May 2020 he obtained his Ph.D. from KU Leuven Research Centre for Information Systems Engineering for his thesis "Decision-Aware Information Systems: A Systems Modelling Perspective Bridging Decisions and Processes". His research has been published in leading journals such as IEEE Transactions on Services Computing, Knowledge-Based Systems, Decision Support Systems, Computer Standards and Interfaces, and Knowledge and Information Systems, and presented at high-ranked conferences, such as the International Conference on Cooperative Information Systems (CoopIS), the International Conference on Business Process Management (BPM), and the ACM Symposium on Applied Computing (ACM SAC). Furthermore, he has a keen interest in technology, history, politics, and the philosophy and history of science. He has an entrepreneurial spirit and enjoys embarking on challenging adventures and projects, as well as travelling and experiencing new people, places, and cultures.

Doctoral Dissertations from the KU Leuven

A full list of the doctoral dissertations from the KU Leuven can be found at:

`www.kuleuven.ac.be/doctoraatsverdediging/archief.htm`.