Enabling Process Management for Loosely Framed Knowledge-intensive Processes

Steven Mertens

Supervisors: Prof. Dr. Frederik Gailly, Prof. Dr. Geert Poels

A dissertation submitted to Ghent University in partial fulfilment of the requirements for the degree of Doctor of Business Economics

Academic year: 2018 – 2019

Copyright © 2019 by Steven Mertens

All rights are reserved. No part of this publication may be reproduced or transmitted in any form or by any means electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without permission in writing from the author.

EXAMINATION BOARD

Prof. Dr. Patrick Van Kenhove (Dean, Ghent University)

Prof. Dr. Amy Van Looy (Secretary, Ghent University)

Prof. Dr. Frederik Gailly (Supervisor, Ghent University)

Prof. Dr. Geert Poels (Supervisor, Ghent University)

Prof. Dr. Fernanda Baiao (Pontifical Catholic University of Rio de Janeiro)

Dr. Melissa De Regge (Ghent University/Ghent University Hospital)

Dr. Niels Martin (Hasselt University)

Dr. Diederik Van Sassenbroeck (AZ Maria Middelaes, Ghent)

PREFACE

“A PhD is like childbirth for the brain”

“How hard can it be? Mind-meltingly, stomach-churningly, sleep-deprivingly difficult”

Emma Jane, The Punch / PerthNow (2011)

If you don't have a PhD: yes, this statement is accurate. A little sympathy wouldn't hurt, you know.

If you're currently doing a PhD: your brain is probably already pregnant. I feel you. Every day it gets a little harder to drag your ailing body out of bed. Dealing with the occasional morning sickness (unrelated to last night's drinking). The prospect of the dreaded third trimester looming (i.e., writing the book and numerous postdoc proposals). Maybe it's not too late to consider a late-term abortion?

If you have a PhD: congratulations! You won... Erm... Well what did you win? I guess you can proudly call yourself a doctor. However, don't bother getting up when someone yells out for a doctor on a plane. They will not be as impressed with the highest academic degree as one might expect.

Now skip past the rest of this preface section, read this book from cover to cover, and only afterwards return here to read the rest. No cheating! You will be quizzed!

Who am I kidding? You're probably not actually going to bother with the rest of this book. Just like if I would be presenting you my newborn, you're going to congratulate me and disappear as soon as you get a whiff of its insides...

How about trying to imagine reading this book from cover to cover? It's over 200 pages of scientific and peer-reviewed work. That's a lot, right? Now imagine having to write such a behemoth. Each chapter/paper needs a problem statement, relevant research questions, a methodology to find answers to those questions, a literature review of the state-of-the-art on the subject and, most importantly, a description of actual contributions to that state-of-the-art. Each chapter/section/paragraph/sentence/word carefully written, criticized by peers, rewritten, criticized by peers, rewritten again, criticized by peers, etc. until everyone involved starts to involuntarily cringe at the mere thought of it. But you can't just start writing. A paper is merely a report of research

that you would've already done by then. The research consists of formulating interesting, relevant and unsolved problems, which you subsequently must tackle head-on – fully aware of the fact that it had probably already crossed the minds of other researchers, who had likely discarded the idea because they considered it an insurmountable problem that only fools would waste their time on. And yet again, you cannot just start formulating interesting, relevant and unsolved problems without being an expert on a certain subject. How do you become an expert? Reading, more reading, even more reading and – you've guessed it – some more reading (my favorite pastime!). Luckily, you can stop when you've learned enough buzzwords to fool other researchers into thinking that you know what you're talking about. Fake it till you make it! After all, we're living in the Trump-era...

"I know words, I have the best words"

Donald J. Trump, 45th president of the United States of America

But before you can start reading up on a certain subject, you need to choose a suitable subject. Something that interests you in a domain with unresolved problems. Problems that are not only practically relevant – your own prerequisite – but also academically relevant. And most importantly, problems for which your particular set of skills could actually be an asset.

"...what I do have are a very particular set of skills; skills I have acquired over a very long [academic] career. Skills that make me a nightmare for people like you [the audience]."

Bryan Mills (played by Liam Neeson) – Taken (2008)

This requires long days of introspective thinking – trying to come up with something that no one else has already done or is currently working on. And then, a eureka-moment! A surefire Nobel prize winning idea! As soon as possible, you meet with your promoters to blow their mind with your amazing idea, only to be met with a lot less enthusiasm than you were expecting. Instead they point out all the (obvious) holes in your idea, while trying to maintain a constructive façade as to not discourage you too much. And this is repeated several times until you find yourself on the couch. Watching House M.D. on TV. Tired of the search. Writing a report on genetic programming for a mandatory course. Thinking that a PhD might not be in your wheelhouse. And then suddenly it hits

you. You like watching House M.D., Dr. Gregory House and his team work in a chaotic manner (lacking any type of process support), your research group covers process management and you're currently writing a report on a technique that can search through a chaotic mess and find good solutions. And so, the seed for this dissertation was planted. It germinated. Its roots took hold. A seedling saw the light. A mighty tree grew. And then they cut it down to print this book.

If you're still reading this: thank you for already making it this far. Mind you, you're still 11 pages removed from the actual first page, but don't let that discourage you. Maybe it will be all worth it. At least I can say that it was worth it for me. The last 6 years have been great, albeit with some unavoidable ups and downs. I got to keep calling myself a student, and sometimes act like one (anyone up for a midweek night out?), even though my bank account kept getting a nice monthly refill. I got to travel and see some beautiful places. And, of course, work with and around some amazing people. First off, I would like to thank my promoters Frederik and Geert for their unwavering support and for creating a great working environment. My jury members Amy, Diederik, Fernanda, Niels and Melissa for all their valuable input and questions. An extra thank you to Melissa for using her connections to get us started in the healthcare domain and to Diederik for his willingness to hear us out, to try to understand our ambitious and overly technical ideas from the get-go, and for providing us with opportunities that would have never been possible otherwise. And then all my other colleagues... Words cannot express my gratitude and the utter pleasure it was spending the last few years with you all, so no point in trying (right?): Aygun, Bambang, Ben, David, Dirk, Dries, Georgios, Gert, Griet, Jan, Jana, Laleh, Machteld, Maria, Martine, Miao, Michael, Miljace, Mohanad, Nadia, Pooyan, Renata, Sven, Tahir, Thiago & Camilla, Thomas, Tom, Xiaji and everyone from the 2nd floor (too much to name). And finally, I would like to thank my family and friends for putting up with me, for their continuous support and for providing me with a timely distraction when it was needed the most.

"Perhaps childbirth is not that bad after all – some people just like to exaggerate."

Steven Mertens – October 2019

TABLE OF CONTENTS

Preface	i
Table of Contents.....	v
List of Figures	ix
List of Tables	x
Abbreviations.....	xi
Summary	xii
1. Introduction	1
1.1. Terminology and Research Context	2
1.2. Problem Statement.....	7
1.3. Research Objectives.....	10
1.4. Research Methodology	15
1.5. Dissertation Structure	17
1.6. Publications.....	19
2. Towards a Decision-aware Declarative Process Modeling Language for Knowledge-intensive Processes.....	23
2.1. Introduction.....	24
2.2. Related Work.....	28
2.3. Research Methodology	29
2.4. Language Requirements	30
2.5. Designing the Perspectives	34
2.5.1. The arm fracture case.....	34
2.5.2. Declare as a foundation.....	35
2.5.3. Extending the functional and control-flow perspectives	39
2.5.4. Adding a notion of time-awareness to the constraint templates	43
2.5.5. Supporting the data perspective	46
2.5.6. Supporting the resource perspective	49
2.6. Integrating the Perspectives.....	52
2.6.1. Integrating the data perspective	53
2.6.2. Integrating the resource perspective	58
2.6.3. DeciClare	61

2.7.	Evaluation	64
2.7.1.	The evaluation set-up	65
2.7.2.	Evaluating the perceived semantic quality	67
2.7.3.	Evaluating the pragmatic quality	68
2.7.4.	Evaluating the language-domain appropriateness	70
2.8.	Positioning of DeciClare in the Research Domain	72
2.9.	Conclusion	74
2.10.	Limitations and Future Work	78
	Research Contributions of Chapter 2	80
3.	Discovering Loosely Framed Knowledge-intensive Processes using DeciClareMiner	83
3.1.	Introduction	84
3.2.	Related Work	86
3.3.	Modeling Healthcare Processes: DeciClare	89
3.4.	Research Methodology	91
3.5.	Discovering Healthcare Process Models: DeciClareMiner	92
3.5.1.	Phase 1: Mining decision-independent rules	93
3.5.1.1.	Leveraged property	95
3.5.1.2.	The iteration principle: optimizations	97
3.5.1.3.	The join criteria	97
3.5.1.4.	The join step	97
3.5.1.5.	Running example	98
3.5.1.6.	The output of the first phase	99
3.5.2.	Phase 2: Mining decision-dependent rules	100
3.5.2.1.	Fitness Function	102
3.5.2.2.	Optimization	103
3.5.2.3.	Running example	104
3.5.2.4.	The output after the second phase	105
3.6.	Evaluation	105
3.6.1.	Dataset	106
3.6.2.	Benchmark creation	107
3.6.3.	Results	108
3.6.4.	Discussion	110

3.7.	Limitations.....	112
3.8.	Conclusions and Directions for Future Research	114
	Research Contributions of Chapter 3	115
4.	Integrated Declarative Process and Decision Discovery of the Emergency Care Process	117
4.1.	Introduction.....	118
4.2.	Research Methodology	121
4.3.	Background.....	123
4.4.	Discovering the Emergency Care Process with DeciClareMiner.....	126
4.4.1.	Extraction.....	126
4.4.2.	Data preprocessing.....	127
4.4.3.	Log preparation.....	130
4.4.4.	Mining a process model.....	133
4.4.5.	Evaluation	135
4.5.	Extracting Knowledge by Modeling the Emergency Care process.....	138
4.6.	Discussion.....	144
4.7.	Conclusion and Future Research	148
	Research Contributions of Chapter 4	150
5.	Operational Support for Loosely Framed Knowledge-intensive Processes.....	153
5.1.	A Generic Framework for Flexible and Data-Aware Business Process Engines.....	155
5.1.1.	Introduction.....	155
5.1.2.	Research Methodology	157
5.1.3.	Problem Statement and Solution Requirements	157
5.1.4.	Data-Aware Declarative Process Enactment Framework.....	160
5.1.5.	Demonstration.....	163
5.1.5.1.	Process Definition: DeciClare	164
5.1.5.2.	DeciClareEngine	164
5.1.6.	Related Work	168
5.1.7.	Conclusion and Future Research	170
5.2.	Comparing Strategies to Generate Experience-based Clinical Process Recommendations that Leverage Similarity to Historic Data.....	172
5.2.1.	Introduction.....	172
5.2.2.	Related Work	174

5.2.3.	Terminology and Formal Problem Definition	176
5.2.4.	Research Methodology	178
5.2.5.	Strategies.....	179
5.2.5.1.	Pre-calculated similarity scorers	180
5.2.5.2.	Positionless similarity scorers.....	180
5.2.5.3.	Variable-position similarity scorers.....	181
5.2.5.4.	Combinated scorers.....	184
5.2.6.	Experiments	185
5.2.6.1.	Experiment 1: comparing the general predictive power over different data sets	185
5.2.6.1.1.	Experimental setup	185
5.2.6.1.2.	Computational results	188
5.2.6.2.	Experiment 2: the relation between predictive power and the size of H	191
5.2.6.2.1.	Experimental setup	191
5.2.6.2.2.	Computational results	191
5.2.7.	Discussion.....	191
5.2.8.	Conclusion and Future Research	196
	Research Contributions of Chapter 5	198
6.	Conclusion and Future Research.....	201
6.1.	Research Results.....	202
6.2.	Research Relevance and Implications	207
6.2.1.	Implications for researchers.....	207
6.2.2.	Implications for process stakeholders.....	207
6.2.3.	Generalizability to other domains.....	210
6.3.	Research Limitations and Future Research	213
	References.....	217
	Appendix A: Arm fracture case description (based on literature)	229
	Appendix B: Original textual DeciClare model of arm fracture case.....	232
	Appendix C: Semi-structured interview protocol (in Dutch)	235
	Appendix D: Final textual DeciClare model of arm fracture case.....	237
	Appendix E: The join step of DeciClareMiner (decision-independent)	248
	Appendix F: The DeciClare templates supported by DeciClareEngine	250

LIST OF FIGURES

Fig. 1. Classification of processes according to the predictability and the data-intensity	3
Fig. 2. The BPM lifecycle (Fig. 1.7 from Dumas et al. (2018b))	4
Fig. 3. The traditional and generalized architecture of a BPMS.....	8
Fig. 4. The design science cycles of this dissertation and their relation to the research objectives (green)	16
Fig. 5. The metamodel of the Declare language	38
Fig. 6. The metamodel of the functional (left) and control-flow (right) perspectives	46
Fig. 7. The metamodel of the proposed data extension	48
Fig. 8. The metamodel of the DMN language for specifying decision logic (based on Figure 51 from (Object Management Group 2016))	48
Fig. 9. The metamodel of the resource extension of the ConDec-R language	50
Fig. 10. The metamodel of the proposed resource extension	52
Fig. 11. The metamodel of the DeciClare language (blue = functional, purple=control-flow, orange = data and green = resource).....	61
Fig. 12. High-level flow diagram of the proposed DeciClareMiner algorithm	93
Fig. 13. Example of the evaluation of the response template	96
Fig. 14. Running example of the first phase (minimal rules in black; non-minimal rules in grey).98	
Fig. 15. Overview of the second phase of the algorithm	101
Fig. 16. Running example of the second phase	104
Fig. 17. Evolution of the intermediate population scores	110
Fig. 18. A high-level overview of the method for process and decision discovery of loosely framed knowledge-intensive processes	126
Fig. 19. An overview of the original database tables (relationships in green and identified activities in blue)	127
Fig. 20. UML class diagram of the conceptual model for the preprocessed data.....	128
Fig. 21. Techniques to structure plain text data fields with example	130
Fig. 22. An example of how the internal representation (preprocessed data) is transformed to a trace.....	131
Fig. 23. The intermediate activities to go from the preprocessed data to an event log.....	131
Fig. 24. The intermediate activities to go from the event log to an integrated process and decision model	133
Fig. 25. The interface of the DeciClareEngine model simulator	137
Fig. 26. The log replay console that contains the original EHR data for the simulated patient	138
Fig. 27. Simplified Declare representation of the mined process model	139
Fig. 28. The results of the evaluation of the mined decision logic by the domain expert	141
Fig. 29. The imperative process model as generated by the Disco-tool	144
Fig. 30. The main interface (left), with added explanations, and activity execution dialog (right)	165
Fig. 31. The results for experiment 2.....	192

LIST OF TABLES

Table 1. The activities used in the arm fracture case	35
Table 2. Seven typical rule patterns for flexible and KiPs	40
Table 3. Seven time patterns to complement the control-flow perspectives	44
Table 4. Three resource patterns relevant to flexible and KiPs	50
Table 5. Six data patterns for KiPs	53
Table 6. Six additional resource patterns linking the resource perspective to the other perspectives	59
Table 7. The control-flow constraint templates of Declare++ and DeciClare	63
Table 8. The data perspective constraint templates of DeciClare	64
Table 9. The resource perspective constraint templates of DeciClare	64
Table 10. A comparison of the languages used in this paper based on the requirements from section 2.4 and their operationalization as patterns from section 2.5 and 2.6	73
Table 11. DeciClare constraint templates used in chapter 3	91
Table 12. Definitions of important concepts	94
Table 13. Mapping of the Apriori algorithm to our problem domain	95
Table 14. Mapping of the genetic approach to our problem domain	101
Table 15. Application of the complete search algorithm for different branching levels	108
Table 16. Results from the mining experiments for different settings	109
Table 17. Explanations of some process mining and/or declarative modeling terminology	125
Table 18. Additional statistics on the event log	133
Table 19. The questions for the domain expert and the underlying purpose behind each of them for the validation	138
Table 20. The four activities used for the evaluation	141
Table 21. The first phase of the framework of a declarative business process engine	161
Table 22. The second phase of the framework of a declarative business process engine	162
Table 23. An overview of the activities and the tier in which they were placed for each snapshot	166
Table 24. The results for experiment 1. For each column, the top performer is formatted in italics and bold and a color-coding was used to represent the best-to-worse ranking (green-yellow-red)	188
Table 25. The bounds for join operation of the condition side of the relation templates	248
Table 26. The bounds for join operation of the consequence side of the relation templates	248
Table 27. The constraint templates supported by DeciClareEngine	251

ABBREVIATIONS

ADR	Action Design Research
AZMM	AZ Maria Middelaers Hospital in Ghent, Belgium
BPE	Business Process Engine
BPM	Business Process Management
BPMN	Business Process Model and Notation
BPMS	Business Process Management System
CMQF	Conceptual Modeling Quality Framework
CP	Clinical Pathway
CPT	Current Partial Trace
DCR	Dynamic Condition Response
DMN	Decision Model and Notation
DSR	Design Science Research
EC	Engineering Cycle
EHR	Electronic Health Record
ER	Engine Requirement
HPT	Historic Partial Trace
HT	Historic Trace
KiP	Knowledge-intensive Process
KPI	Key Performance Indicator
LR	Language Requirement
LTL	Linear Temporal Logic
MTL	Metric Temporal Logic
RC	Research Cycle
RQ	Research Question
SQL	Structured Query Language

SUMMARY

A business process consists of the collection of activities and resources, including people, that are needed to achieve a business goal of an organization. Processes are the foundation of every organization. Therefore, it is important that they are actively supported and managed, so that the organization can continuously maintain or improve its efficiency and efficacy. Research has proposed many ways to offer this support and enable the management of business processes. However, the inspiration for this dissertation came from the observation that the currently available options do not sufficiently cover all types of processes. Loosely framed knowledge-intensive processes are one of the types of processes that had been left out. These processes are characterized by their numerous valid process variants and their reliance on knowledge workers to apply their knowledge to decide on a suitable process variant that fits the context of a specific process execution. Consider, for example, the diagnosis and treatment of patients in hospitals. The physicians are the knowledge workers who apply their medical knowledge to decide on a path towards a diagnosis and the subsequent treatment of the patient based on his/her symptoms, medical history, preferences and responses to intermediate treatments as well as the results of diagnostic tests and the overall situation.

There are many different flavors of business process management. In this dissertation, the goal was to develop a process-aware business process management (IT-)system for loosely framed knowledge-intensive processes. Based on the minimum requirements for such a system, three objectives were defined:

Objective 1: The development of a declarative process modeling language that suits knowledge-intensive processes (chapter 2).

Objective 2: The development of a method for process and decision discovery of loosely framed knowledge-intensive processes (chapter 3) and a corresponding mining technique that can mine a process and decision model from an event log (chapter 4).

Objective 3: The development of a business process engine (section 5.1), which can automatically interpret a process and decision model, and a process analytics overlay (section 5.2), which can offer guidance to the process actors during the process execution.

The result of each of these three objectives is the core of a component of the business process management system of for loosely frame knowledge-intensive processes. Proof-of-concept implementations have been made and evaluated for each of these components. The application domain throughout this dissertation was the healthcare domain, and more specifically, the diagnosis and treatment of patients in the emergency department. We started with an artificial, but realistic, case about patients that arrive in the emergency room with suspected arm fractures and later did a case study in the emergency department of a real hospital, using data from their patient files. The last chapter concludes the dissertation, discusses the generalizability of the results to other domains, lists the research contributions and implications for researchers and practitioners, and outlines possible future research opportunities.

1

Introduction

1.1. TERMINOLOGY AND RESEARCH CONTEXT

This PhD addresses shortcomings that relate to the process orientation and support in settings that house loosely framed knowledge-intensive processes. Before diving into the problem, we will first briefly describe the general domain, the characteristics of the target processes and the architecture of the tool support.

Business Process Management (BPM) is the field that researches how consistent outcomes can be achieved and continuously improved by managing the way work is performed in an organization (Dumas et al. 2018; Trkman 2010). The way of performing work is captured in the concept of a **business process**, which is a collection of inter-related events, activities, decision points and actors that collectively deliver value to customers or realize a business goal (Guha et al. 1993; Strnadt 2006; Weske 2012). Therefore, BPM is defined as a body of concepts, methods, techniques and tools to support the design, configuration, discovery, analysis, redesign, execution and monitoring of business processes (Dumas et al. 2018).

A possible way of classifying business processes is according to their predictability¹ (van der Aalst 2013; Di Ciccio, Marrella, et al. 2015). First, a **tightly framed** process is always executed in just one of a few predefined, consistent and unambiguous ways (e.g., the mass production of electronics). When the process is less structured it is called **loosely framed or flexible**, which entails that a process is executed in a large, but finite and predefined, number of ways (e.g., the treatment of patients in a hospital). Next, a process is **ad hoc framed** if it is typically executed starting from a predefined process model, but ad hoc changes occur frequently during the execution. This leads to the models being used only once or perhaps a few times before being discarded or changed (e.g., project planning). Lastly, **unframed** processes are processes where each execution is unique and generally impossible to predefine (e.g., groupware systems).

¹ Predictability of the actual processes, not of models made of the processes!

Fig. 1. Classification of processes according to the predictability and the data-intensity

This classification according to the predictability of processes can be further complemented with a data-intensity dimension (see Fig. 1). The data is linked to the choice for a certain the process variation, and therefore, becomes more and more relevant as the predictability decreases. The scale ranging from tightly framed to unframed processes typically corresponds to processes becoming more and more data-intensive (gray diagonal in Fig. 1) (van der Aalst 2013). Processes situated in the lower left corner of Fig. 1 have few variations and no or some simple decisions (e.g., manufacturing processes), while processes in the upper right corner have many variations and many complex decisions (e.g., the diagnosis and treatment of patients in a hospital). Off-diagonal processes are certainly possible, but rather the exception than the rule. Processes situated in the lower right corner have few variations but complex decisions (e.g., the loan approval process in a bank), while processes in the upper left corner have many variations but no or just a few simple decisions (e.g., a ‘naïve’ theme park experience, where all rides are an option all the time).

The dark area in Fig. 1 corresponds to what are defined as **Knowledge-intensive Processes** (KiPs). These are business processes whose conduct and execution are heavily dependent on knowledge workers performing various interconnected knowledge-intensive decision-making tasks (Di Ciccio, Marrella, et al. 2015; Reichert and Weber 2012). KiPs are genuinely knowledge-, information- and data-intensive and require substantial flexibility at design- and run-time. Many ad hoc framed and unframed processes can also be regarded as KiPs but are out of scope of this

dissertation, because the lack of framing (i.e., cannot predefine the activities and process variations) means that they require a different management approach.

The application of BPM in organizations can be mapped to the phases of the **BPM lifecycle** (see Fig. 2), which encompasses a range of methods and tools to identify processes and to manage individual processes (Dumas et al. 2018). It starts with the **identification** of the processes related to a specific business problem. In the next phase, process **discovery**, the current state of the relevant identified processes is documented in the form of (as-is) models. Issues in the current state will subsequently be identified, documented and quantified as part of the process **analysis** phase. In the phase that follows, changes to counter the issues are identified and compared. The most promising change options are combined to **redesign** the processes. The proposed changes will then need to be **implemented** in the form of organizational changes and/or process automation. Once the redesigned processes are up and running, relevant data will be collected and analyzed to evaluate the impact of the changes on the identified process issues during the **monitoring and controlling** phase. And it

Fig. 2. The BPM lifecycle (Fig. 1.7 from Dumas et al. (2018b))

does not end here. New issues may arise in the redesigned or other processes of the organization. Therefore, the BPM lifecycle is a cycle that needs to be repeated on a continuous basis.

Starting from the process discovery phase onwards, a representation is needed of the identified processes. **Business Process Modeling** entails the explicit representation of business processes by describing its activities, events/states, and control-flow logic (Recker et al. 2009). Such representation is called a process model. These models can subsequently be used as a way of documenting a process, as a way of representing the as-is and to-be situations, as a means to check conformance and/or as a precise schedule for the execution of the process (van der Aalst 2011a).

The discovery of the processes can be performed manually, for example, by way of documentation analysis, interviews with process actors and/or on-site observation. However, the daily operation of organizations is often supported by information systems, which contain valuable data related to the real-life execution of the processes. This data can be used to reconstruct the processes. The domain of **Process Mining** is concerned with the development and application of (semi-)automated tools that return as-is process models based on the process data extracted from information systems (van der Aalst et al. 2011). However, a requirement for these tools to do their job properly is the availability of digitalized data, which is not always available, complete or of high quality.

The process data that is used as input for the process mining tools describes **process instances**. A process instance is one specific execution of a business process and consists of a list of **process events**. Each event represents a relevant occurrence during the execution of a process instance. We make a distinction between activity and data events. The former represents, for example, the start, end or failure of the execution of an activity (e.g., take medical history). The latter represents the timestamp of when a data value is added, manipulated or deleted (e.g., 'blood pressure of 140/90' written down in the electronic health record). The chronological list of all events relating to one process instance is called a **trace**. In healthcare, a trace aligns with what is called an episode of care for one patient (e.g., a chronological list of all activities and data related to the registration, triage,

diagnosis and treatment of a certain patient). A set of traces for one process (e.g., a collection of traces of all patients that received emergency care) as logged by the IT systems of the organization is called an **event log**. To make such an event log, data first needs to be extracted and combined from the data sources that span the information system of the organization. The resulting data should subsequently be filtered and preprocessed into an event stream format.

The process implementation phase of the BPM lifecycle covers two aspects: organizational change management and process automation (Dumas et al. 2018). In this dissertation, the focus will be on the side of process automation. More specifically, process automation supported by the use of a **Business Process Management System (BPMS)** (a.k.a. Process Aware Information System). A BPMS is a generic software system that coordinates the enactment of business processes involving people, applications and/or information sources based on process models (van der Aalst 2009; Weske 2012). This definition was later expanded to define the BPMS as a more general collection of components that serve the goal of managing business processes (traditional BPMS in Fig. 3) (Vassilakopoulou et al. 2015). The components of a BPMS are:

- **Business Process Design**-component: allows users to view, create and modify the business process models used by the BPMS. This component relates to the process discovery and redesign phases of the BPM lifecycle.
- **Business Process Engine**-component (BPE): is responsible for the correct enactment of process models. This component relates to the process implementation and, to a lesser extent, the controlling and monitoring phases of the BPM lifecycle.
- **Business Process Analytics**-component: enables analysis of the process models, the executing/historic process instances and simulations. Typically, the results of these analyses are presented as a process dashboard or in reports. This component relates to the process analysis and the controlling and monitoring phases of the BPM lifecycle.

- **Business Process Monitoring**-component: monitors the execution of running process instances. This is often in terms of Key Performance Indicators (KPIs), but more advanced features such as predictions of potentially unwanted future behavior or outcomes are also possible. This component relates to the process controlling and monitoring phase of the BPM lifecycle.
- **Business Process Exception Handling**-component: is responsible for how the BPMS handles process exceptions (i.e., deviations from the given process models) in real-time. This component does not directly relate to any of the BPM lifecycle phases but can be regarded as a fail-safe to handle new or unidentified process issues. The invocation of this component should be the starting point for a new iteration of the lifecycle.
- **Business Process Simulation**-component: offers simulations based on a given process model and stochastic functions for the other parameters. It can be used to analyze the dynamic behavior of the process over time. This component relates to the process analysis and redesign phases of the BPM lifecycle.

Process discovery is often already being applied in the context of a traditional BPMS-components and serves the goal of managing business processes as was put forward as the idea behind the original expansion of the BPMS responsibilities. As a result, we will also include a **Business Process Discovery**-component, which supports methods for process (and decision) discovery from a given event log. The resulting model can be used as input for the other components. Fig. 3 presents the components of a traditional (extended) and of a generalized BPMS that includes this extra component.

1.2. PROBLEM STATEMENT

This dissertation started from an observed lack of a BPM framework for loosely frame KiPs. The characteristics of these processes have proven to be a formidable obstacle to the successful adoption of BPM to those settings (Di Ciccio, Marrella, et al. 2015). On the one hand, their loosely framed

Fig. 3. The traditional and generalized architecture of a BPMS

nature leads to processes where deviations and variations are the norm rather than the exception. Therefore, most, if not all, of these process variations need to be accounted for one way or the other by the framework to be able to provide the necessary **flexibility**. On the other hand, their knowledge-intensive nature stems from the need to choose a process variation for each process instance. Each variation applies to a certain setting, and the process actors, also called knowledge workers, need to decide which one they find appropriate for the process instance at hand. The immense number of process variations means that the amount of knowledge needed to distinguish between them is also immense.

The knowledge of knowledge workers is often referred to as **tacit knowledge**, in that it primarily resides in some implicit form in their memory (Di Ciccio, Marrella, et al. 2015). Loosely framed KiPs often also have a **dynamic** component (Lenz and Reichert 2007). The knowledge needed to execute the process as well as the acceptable process variations can evolve over time (e.g., medical procedures can change due to new insights or technological advancements). This further exacerbates the problem of knowledge remaining tacit, because efforts to document it are confronted with a heavy struggle to stay up-to-date. This also applies to the knowledge workers themselves. Not everyone is as aware of every novelty, which potentially leads to differences in how each would handle a certain situation. Additionally, each knowledge worker typically possesses a subset of the

total tacit knowledge dependent on their role in the process. Therefore, a complete picture of the process knowledge is only possible by combining all pieces of knowledge, like pieces of a puzzle, without a guarantee that there will not be any conflicts. This also causes personnel turnover to be potentially very disruptive, even disastrous, because it is unclear what knowledge each of the departing knowledge workers possessed. That knowledge is inevitably lost and there is no way to determine whether potential replacements can sufficiently make up for this loss.

Process modeling provides a medium for transforming tacit process knowledge to explicit knowledge, because a process model encapsulates the process knowledge and makes it available to all stakeholders. This also enables further analysis to correct, redesign and optimize the process. However, the need for flexibility, the lack of transparency and the dynamic nature of these processes also hinder the adoption of process modeling techniques. In the medical sector, they have tried to sidestep these issues with the use of clinical pathways (CPs) and similar concepts like clinical guidelines, critical pathways, integrated care pathways and care maps. Although the exact definition of what they entail can vary from paper to paper (Vanhaecht et al. 2006), the general consensus seems to be that they are best practice schedules of medical and nursing procedures for the treatment of patients with a specific pathology (e.g., breast cancer). CPs can be considered process models, but real-life situations are typically much more complex. Comorbidities and practical limitations (e.g., resource availability) can prevent CPs from being implemented as-is. The real process also comprises more than just the treatment of patients, because activities to try and reach a diagnosis and other practicalities should be considered as well. A CP is therefore a form of high-level model of one idealized process variation. Modeling every process variation explicitly is easy for tightly framed processes, where there are only a couple of variations possible. This referred to as **imperative modeling** (e.g., BPMN) and is used to explicitly specify all allowed behavior. However, the numerous acceptable process variations of loosely framed processes make this approach less suitable for loosely framed processes.

The lack of a suitable process modeling language also severely inhibits the development of any kind of **operational process support** for these processes. The BPE-component of a BPMS requires a process model to provide users with information and support on how they can proceed during the execution of a process. In the current situation, process support for loosely framed KiPs typically ranges from non-existent to rudimentary at most. A process model is almost never available to serve as a foundation to offer support or assistance. Therefore, knowledge workers are currently left to their own devices, but they are still expected to execute the process instances in a satisfactory manner. This puts a lot of responsibility and pressure on the knowledge workers. The result is a process that scores very bad when it comes to **transparency**. This lack of transparency is also a barrier to analysis and optimization, which in turn can lead to questions about the unmeasurable efficiency.

1.3. RESEARCH OBJECTIVES

The general objective of this dissertation is the development of a BPMS for loosely framed KiPs. It should support the different phases of the BPM lifecycle as described in section 1.1. In the scope of this dissertation, we will focus on the development of the main components of such a system: the Process Designer-, Discovery-, Engine- and Analysis-components.

General objective: The development of a BPMS for loosely framed KiPs

The Process Designer-component requires a process modeling language that suits loosely framed KiPs. **Declarative modeling languages** (e.g., Declare) take on the opposite viewpoint of imperative languages: the outside-in approach. They describe the rules and constraints by which the process variations must abide (Goedertier et al. 2015). The unwanted process variations are prohibited based on these rules and constraints, while all other variations are allowed implicitly and without having to enumerate each valid variations (van der Aalst et al. 2009).

Although the basic principles behind declarative process modeling languages are well suited to model loosely framed KiPs (van der Aalst 2013; Goedertier et al. 2015; Rovani et al. 2015), the suitability of a specific declarative language also depends on the availability and expressive power of each of the necessary business process modeling perspectives (Curtis et al. 1992). First, the **functional** (i.e., the available activities/events and their instances) and **control-flow** (i.e., restrictions on when activities/events occur) perspectives are the core of a process modeling language. For declarative languages, these two perspectives provide an innate support for the ‘flexibility by design’-principle (Schonenberg et al. 2008) that is essential to model the flow of activities in loosely framed KiPs. Second, the ‘knowledge’ in KiPs alludes to the implicit or explicit knowledge needed by the process workers to execute the process. KiPs are governed by many rules and/or decisions processes (Borrego and Barba 2014; Di Ciccio, Marrella, et al. 2015) and insight into these decisions is at least as important as what activities are available for execution. Modeling a decision process comes down to linking combinations of data values to some sort of outcome, which in turn requires a **data** perspective (a.k.a. information perspective). Third, resources play a vital role in loosely framed KiPs. Often, the execution of certain activities from the functional and control-flow perspectives depends on the availability of certain resources and those resources have certain properties expressed in data values (e.g., availability, speed, cost, quality, etc.) that influence the decisions in the data perspective of the process model. So, some form of **resource** perspective is also needed.

There exist several declarative process modeling languages, with the most popular being Declare (van der Aalst et al. 2009; Pesic 2008) and DCR Graphs (Mukkamala 2012). The original definition of these languages addresses the functional and control-flow perspectives. Since then, several extensions have been proposed to increase the expressive power of these languages by adding additional functional and control-flow elements or adding new perspectives. However, none of the previously proposed declarative languages and extensions combines all the necessary perspectives for loosely framed KiPs.

Objective 1: The development of a declarative process modeling language that suits loosely framed KiPs, which lays the foundation for the Process Designer-component of the BPMS

A suitable process modeling language for loosely framed KiPs creates an opportunity to gain insight into these processes. However, the difficulty now lies in finding the right approach to create the process models, as most of the process knowledge is of a tacit nature. Traditionally, this is done by way of interviews with process actors and other domain experts. This is straightforward when there is little knowledge to extract, but otherwise can quickly become a very difficult, effort-intensive, time-consuming and error-prone task for both the modeler and the process actors. The inclusion of a process mining technique in the Process Discovery-component can provide an appealing alternative. It can offer a direct reflection of how things are actually done (i.e., an as-is process model), instead of an idealized description from interviews. Traditional process mining techniques automatically detect control-flow patterns in a given dataset and turn these into a consistent process model that accurately reflects the process execution data.

Most process mining techniques assume an imperative viewpoint on the process. Recently, several declarative process mining techniques have been proposed (vanden Broucke et al. 2014; Burattin et al. 2012; Di Ciccio et al. 2016; Di Ciccio, Schouten, et al. 2015; Debois et al. 2017; Maggi 2013, 2014; Maggi et al. 2012, 2018; De Smedt et al. 2015). The problem is that each of these techniques returns a process model written in one of the existing declarative process modeling languages or its extensions, and hence, not in the process modeling language that will be the result of objective 1. So, a new process mining technique is needed for the language definition of objective 1, which potentially could incorporate pieces of the existing declarative process mining techniques. In the context of loosely framed processes we are not just interested in mining the functional and control-flow perspectives, but also in a data perspective that contains the decision logic that governs the functional and control-flow perspectives. The automatic mining of a resource perspective is of lesser importance, because most organizational IT-systems do not log the data required (e.g., availabilities, authorization and usage), and will be regarded as future research.

The application of a process mining technique, which assumes that a satisfactory event log is available, is just one part of a larger method(ology) for process discovery. This method(ology) consists of the necessary steps to gather data, organize and process this data, the preparation of an event log that is aligned with the input requirements of the chosen process mining technique, the application of the process mining technique, a subsequent evaluation of the discovered model and iterative repetitions needed to obtain a satisfactory model. Two such methodologies have been proposed for imperative process discovery projects (van Eck et al. 2015; Rebuge and Ferreira 2012), which focus on the functional and control-flow perspectives. However, a methodology for declarative process and decision discovery is still missing.

Objective 2: The development of a method for process and decision discovery of loosely framed KiPs for the Process Discovery-component of the BPMS, and a corresponding mining technique that can mine a process and decision model from an event log

An accurate process model enables a deeper level of management and analysis. The question becomes how to use and leverage the model for the benefit of the process stakeholders. Declarative process modeling languages are known to be hard to understand by human actors (Haisjackl et al. 2014). The innate complexity of loosely framed KiPs only exacerbates this further. So, there is no point in putting a printed version of the process model, perhaps already redesigned or optimized, in the hands of the process actors and expecting them to enact it. However, computers are not bothered as much by this complexity, as long as the modeling language allows for an unambiguous interpretation of the models. They are much better at processing this type of information and certainly in these quantities. This is where the operational components of the BPMS come into play. The foundational architecture for traditional BPMS, called a workflow management system, was published in 1995 (Hollingsworth 1995). A process model serves as the input for such system and its components enable monitoring, management, analysis, optimization and simulation. Traditional BPMSs expect an imperative process model as input. These models have clearly defined paths that can be followed, whereas declarative process models just provide the rules for such valid paths but

lack any sort of explicit definition. This difference has a profound impact on the requirements of some of the components of a BPMS.

For example, a traditional BPE simply must follow the explicitly defined paths of the given imperative model to determine the next and future activities. In contrast, a declarative process model does not define any explicit paths. The requirements for a BPE that executes declarative process models are therefore different and more complex, compared to a traditional BPE. A declarative BPE will first have to calculate which parts of the model apply to the context at hand. This filtered model subsequently enables the calculation of the activities that can be executed next, the activities that must eventually be executed and those that are prohibited either at specific moments or during the remainder of the process instance. It is of paramount importance that every path allowed/prohibited by the declarative process model is also allowed/prohibited by the BPE, to preserve the freedom to operate of the knowledge workers and prevent divergence from the model. When done correctly, this can significantly reduce the size of the set of rules that process actors get exposed to at one time, and thus, minimize the impact of the human understandability issues associated to declarative process models. The rules of the model that led to an activity being placed into one of the aforementioned categories can be made available to the process actors on-demand, exposing them to merely those pieces of the model that they require to accurately interpret it.

The intended BPE should be able to automatically interpret a given declarative process model, which relieves the process actors of this difficult and time-consuming task but offers little guidance when presented with the choice of multiple valid process variations. One way of offering guidance is to provide the process actor with analytical information on the available variations that can be used to rank them. This functionality is arguably not a part of the BPE-component itself, but rather of the Process Analytics-component of a BPMS as an additional layer on top of the BPE. The combination of these two subcomponents should provide rich operational support to process actors during the execution of a process instance. The BPE-component ensures that the execution conforms

with the explicit knowledge of the given process and decision model, while the Process Analytics-component is used to offer insight and guidance.

Objective 3: The development of a BPE- and Process Analytics-component for the BPMS that can offer support and guidance to process actors during the process execution

In summary, this dissertation will focus on the development of the core functionality of a generalized BPMS for loosely framed KiPs: the Process Designer-, Discovery-, Engine- and Analytics-components.

1.4. RESEARCH METHODOLOGY

The **Design Science Research** (DSR) methodology (Hevner et al. 2004; Peffers et al. 2007) was applied in this research project, with an **Action Design Research** (ADR) cycle mixed in. DSR can be used to create and evaluate design artifacts such as constructs, models, methods and instantiations with scientific rigor (Hevner et al. 2004). Each artifact should solve one or more problems relevant to practice while the development and evaluation process should contribute new knowledge to the state of the art. DSR is inherently iterative and the creation of one artifact often requires the creation of nested artifacts. Therefore, it is best visualized using nested design cycles (Wieringa 2009). We make a distinction between two types of cycles: engineering and research cycles (Wieringa and Heerkens 2006). An engineering cycle (EC) consists of the necessary steps to design and evaluate an artefact to solve a well-defined problem. A research cycle (RC) is responsible for resolving a research related issue such as establishing the state of the art for a problem, finding and adapting related techniques, etc. Fig. 4 presents an overview of the design cycles of this dissertation.

The general objective from the previous section was translated to the main engineering cycle (EC1), which is responsible for developing a BPMS for loosely framed KiPs. This cycle had three smaller engineering cycles (EC2, EC3 and EC5).

EC2 was the translation of research objective 1, the development of a process modeling language for loosely framed KiPs and contained two smaller research cycles (RC1 and RC2). Literature

Fig. 4. The design science cycles of this dissertation and their relation to the research objectives (green)

already established that a modeling language that takes on a declarative viewpoint is preferable to one that takes on an imperative viewpoint. Therefore, RC1 was concerned with the identification and assessment of the existing declarative process modeling languages. The aspects of these languages that were deemed useful in the context of loosely framed KiPs were subsequently combined and extended in the continuation of EC2. The resulting process modeling language, called DeciClare, was then evaluated in RC2.

EC3 was the translation of research objective 2, the development of a method for process and decision discovery for loosely framed KiPs with a model written in the language developed in EC2 as output, and contained an engineering cycle (EC4) and a research cycle (RC3). We chose to apply ADR as methodology for EC3, which is a specific type of design science research for the design of research artifacts that explicitly provide theoretical contributions to the academic knowledge base, while simultaneously solving a practical problem (Sein et al. 2011). Creating a process and decision

model manually is unrealistic in practical settings for this type of processes. Therefore, an automated process and decision mining technique was first developed in EC4, which was a normal design science engineering cycle to start the ADR-cycle. The ADR-cycle itself focused on the development of the general discovery method, while simultaneously evaluating the method and mining technique by applying them to a real-life setting (RC3).

EC5 was the translation of the first part of research objective 3, the development of the robust BPE-component of a BPMS for loosely framed KiPs, and contained a smaller engineering cycle (EC6). A BPE that can merely support the execution of a declarative model is not very supportive, because loosely framed KiPs typically have a number of activities that can be executed next at any time during the execution of a process instance. Therefore, EC6 was the translation of the second part of research objective 3, the development of a recommendation generator for the BPE, and contained three smaller research cycles (RC4-RC6). During RC4 and RC5 we identified and assessed strategies from the operations research and predictive process monitoring domains that could be used to generate next-activity recommendations. The strategies that were deemed applicable, were subsequently implemented, evaluated and compared in RC6.

1.5. DISSERTATION STRUCTURE

This dissertation follows the structure of a paper-based dissertation, consisting of six chapters.

Chapter 1: Introduction

This chapter introduces the domain of process management, the targeted processes, the problems to be addressed, the research objectives and the chosen approach to achieve these objectives.

Chapter 2: Towards a Decision-aware Declarative Process Modeling Language for Knowledge-intensive Processes

This paper is the result of the second iteration of RC1 and EC2, and the first iteration of RC2. It describes the development of a process modeling language for loosely framed KiPs.

Chapter 3: Discovering Loosely Framed Knowledge-intensive Processes

This paper is the result of EC4. It describes the development of an automated process and decision discovery technique for loosely framed KiPs.

Chapter 4: Integrated Declarative Process and Decision Discovery of the Emergency Care Process

This paper is the result of EC3. It describes the development of a method for creating a process and decision knowledge base for loosely framed KiPs. The ADR-cycle combines the development of this method with the application of the process and decision discovery technique on real data from the emergency medicine department of a hospital (RC3).

Chapter 5: Operational Support for Loosely Framed Knowledge-intensive Processes

This chapter consists of two related papers. The first paper is the result of EC5 and describes the development of a data-aware BPE that supports the execution of a declarative process model. The second paper is the result of the first iteration of EC6 and RC4-6 and describes the evaluation and comparison of forty strategies to generate next-activity recommendations. The combination of these two subcomponents produces a fully operational BPE, which ensures that the process is executed in conformance to a given process model while also providing recommendations that support process actors during the execution of a process instance.

Chapter 6: Conclusion and Future Research

This chapter concludes the dissertation, discusses the generalizability of the results to other domains, lists the research contributions and implications for researchers and practitioners, and outlines possible future research opportunities.

1.6. PUBLICATIONS

Publications in peer-reviewed international journals

- Mertens, S., Gailly, F., Poels, G.: *Discovering healthcare processes using DeciClareMiner*. Heal. Syst. - Spec. Issue SIG-Health Des. Innov. Impact Healthc. IT. 7, 195–211 (2018). (EC4)
- Mertens, S., Gailly, F., Poels, G.: *Towards a decision-aware declarative process modeling language for knowledge-intensive processes*. Expert Syst. Appl. 87, 316–334 (2017). (EC2-RC1-RC2)

Publications under review in peer-reviewed international journals

- Mertens, S., Gailly, F., Van Sassenbroeck, D., Poels, G.: *Integrated Declarative Process and Decision Discovery of the Emergency Care Process*. (2019). (EC3-RC3)

Publications in peer-reviewed international conference proceedings

- Mertens, S., Gailly, F., Van Sassenbroeck, D., Poels, G.: *Comparing strategies to generate experience-based clinical process recommendations that leverage similarity to historic data*. In: 2019 IEEE International Conference on Healthcare Informatics (ICHI). IEEE (2019). (EC6-RC4-RC5-RC6)
- Mertens, S., Gailly, F., and Poels, G.: *A Generic Framework for Flexible and Data-Aware Business Process Engines*. In: H. Proper and J. Stirna (eds.) FAISE Workshop at CAiSE'19. LNBIP, vol. 349, pp. 201–213, Springer, Cham (2019). (EC5)
- Mertens, S., Gailly, F., Poels, G.: *Supporting and assisting the execution of flexible healthcare processes*. In: Proceedings of the 9th International Conference on Pervasive Computing Technologies for Healthcare. pp. 334–337. IEEE Xplore, Istanbul, Turkey (2015). (EC1)

- Mertens, S., Gailly, F., Poels, G.: *Enhancing declarative process models with DMN decision logic*. In: Enterprise, Business-Process, and Information Systems Modelling at BPMDS/EMMSAD'15, held at CAiSE'15. LNBIP, vol. 214. pp. 151–165. Springer, Stockholm, Sweden (2015).
(EC2)
- Mertens, S.: *Supporting and assisting the execution of knowledge-intensive processes*. In: Loucopoulos, P., Nurcan, S., and Weigand, H. (eds.) Proceedings of the Doctoral Consortium of CAiSE'15, vol. 1415. pp. 10–18. CEUR-WS, Stockholm, Sweden. (2015).
(EC1)
- Mertens, S., Gailly, F., Poels, G.: *Generating business process recommendations with a population based meta-heuristic*. In: BPM'14 Workshops, DeMiMoP'14. LNBIP, vol. 202. pp. 516–528. Springer, Haifa, Israel (2014).
(EC6)

Other conferences and workshops

- Mertens, S., Gailly, F., Poels, G.: *Enacting loosely framed and knowledge-intensive processes*. 5th Belgian Process Mining Research Day in Leuven, Belgium (2018).
(EC5-EC6)
- Mertens, S., Gailly, F., Poels, G.: *Mining the process of a real emergency medicine department*. 4th Belgian Process Mining Research Day in Ghent, Belgium (2017)
(EC3-RC3)
- Mertens, S., Gailly, F., Poels, G.: *Discovering and Modeling Healthcare Processes: DeciClare*. SIG Health workshop at the 36th International Conference on Information Systems in Dublin, Ireland (2016).
(EC4)
- Mertens, S., Gailly, F., Poels, G.: *A Combined Approach to Declarative Process Mining and Decision Mining*. 3th Belgian Process Mining Research Day in Hasselt, Belgium (2016).
(EC4)

Chapter 2 describes EC2, RC1 and RC2 (see section 1.4).

Reference:

Mertens, S., Gailly, F., Poels, G.: *Towards a decision-aware declarative process modeling language for knowledge-intensive processes*. Expert Syst. Appl. 87, 316–334 (2017).

2

Towards a Decision-aware Declarative Process Modeling Language for Knowledge-intensive Processes

2.1. INTRODUCTION

A business process is a “collection of inter-related events, activities and decisions points that involve a number of actors and objects, and that collectively lead to an outcome that is of value to at least one customer” (Dumas et al. 2018). Business process modeling (BPM) is the practice of creating models that represent business processes (Dumas et al. 2018; Weske 2012). These models can subsequently be used as a way of documenting a process, as a way of representing the as-is and to-be situations, as a means to check conformance and/or as a precise schedule for the execution of the process (Dumas et al. 2018). Business process modeling is being applied in a variety of industries (e.g., manufacturing, insurance, utilities, retail, telecommunications, finance, etc.). As a result of the diversity in uses and target domains, a multitude of process modeling languages exist (van der Aalst 2013; Goedertier et al. 2015; Krogstie 2013; Lu and Sadiq 2007). Gantt and flow charts were part of the first wave and have since given way to modern-day techniques like BPMN (White 2004), UML activity diagrams (Object Management Group 2015) and Declare (van der Aalst et al. 2009; Pesic 2008).

A possible way of classifying business processes is according to their *predictability* (van der Aalst 2013; Di Ciccio, Marrella, et al. 2015). First, a *tightly framed* process is always executed in just one of a few predefined, consistent and unambiguous ways (e.g., the mass production of electronics). When the process is less structured it is called *loosely framed*, which entails that a process is executed in a large, but finite and predefined, number of ways (e.g., the treatment of patients in a hospital). Next, a process is *ad hoc framed* if it is typically executed starting from a predefined process model, but ad hoc changes occur frequently during the execution. This leads to the models being used only once or perhaps a few times before being discarded or changed (e.g., project planning). Lastly, *unframed* processes are processes where each execution is unique and generally impossible to predefine (e.g., groupware systems). The scale ranging from tightly framed processes to unframed processes corresponds to processes becoming more and more knowledge-intensive (van der Aalst 2013). Although there exist strongly framed processes that can be

considered knowledge-intensive (e.g., the loan approval process in a bank), these are more the exception rather than the rule. Di Ciccio, Marrella, et al. (2015) define *knowledge-intensive business processes (KiPs)*, in general, as follows:

“Processes whose conduct and execution are heavily dependent on knowledge workers performing various interconnected knowledge-intensive decision-making tasks. KiPs are genuinely knowledge-, information- and data-intensive and require substantial flexibility at design- and run-time.”

The former part of this definition references the complex nature of these processes and the unmistakable human factor in all of this. The execution of the process is completely dependent on *knowledge workers* (e.g., doctors), who each use their specific knowledge to contribute to the process. This knowledge is a form of processed information (i.e., understood patterns) and information (i.e., detected patterns) can be defined as data values (i.e., raw data) with a certain meaning or purpose (Rowley 2007). So, in essence, modeling knowledge requires the modeling language to support at least a *data* perspective (i.e., the available data records, attributes, values and their relationships). The latter part of the definition references the loosely framed to unfixed nature of most KiPs. A knowledge worker (i.e., part of the *resource* perspective with the available resources and their relationships) requires a significant amount of freedom to operate, so that he or she can apply their knowledge to specific contexts. This freedom can be incorporated at design-time, at run-time or both (Reichert and Weber 2012). For example, a model can explicitly specify a decision point and a specific process variation for each decision outcome at design-time or even leave these underspecified to be determined at run-time. Basically, this part of the definition indicates that a *functional* (i.e., the available activities/events and their instances) and *control-flow* perspective (i.e., restrictions on when activities/events occur) are integral parts of a modeling language for KiPs to be able to cater to this need for flexibility.

The classification can now be translated to a classification based on the relative importance of the functional, control-flow, resource and data perspectives to the process. For tightly framed

processes the focus is mainly on precise definition of the functional and control-flow perspectives, and in some cases also on the resource perspectives (Di Ciccio, Marrella, et al. 2015). However, for loosely framed processes the data perspective is at least of equal importance as the other perspectives, because it is an essential requirement to model the knowledge that governs the other perspectives (Di Ciccio, Marrella, et al. 2015). The other perspectives are also of importance as demonstrated by the need for flexibility in the functional and control-flow perspectives in the definition of KiPs, while knowledge workers themselves are part of the resource perspective and knowledge about resources can heavily influence their decisions. The languages that currently target loosely framed process (e.g., Declare, DCR graphs, GLIF, Asbru, PROforma, etc.) do not yet sufficiently address all of these perspectives together. This is the research gap that this paper will try to tackle. The relative importance of the data and resource perspectives increases even more for ad hoc framed processes and ultimately dwarfs the other perspectives in the context of unframed processes.

The knowledge of process workers in KiPs guides the decision-making regarding which process variation is appropriate in a certain situation (e.g., how doctors diagnose and treat patients). The *tacit* nature of this knowledge is a typical symptom of the process being in a state of flux as the domain knowledge itself changes with time (Krogstie 2013; Lenz and Reichert 2007). The process workers cannot keep up with the complexity of the constant stream of new knowledge, which results in a situation where everyone has a different subset of the total, possibly even out-of-date, knowledge. Additionally, nobody knows exactly what the other process workers know. This tacit knowledge phenomenon is an important obstacle when attempting to accurately model these processes (Ferreira and Ferreira 2006). Stimulating and managing the knowledge conversion processes in order to externalize the tacit knowledge (i.e., linking sets of data values to certain outcomes) is an important part of improving these processes. Firstly, the explicit knowledge can be used for explaining decisions that were made (Object Management Group 2016). Secondly, making process knowledge explicit increases the transparency of the process to all stakeholders, and hence

facilitates communication of process changes or new insights as the process evolves (Object Management Group 2016). Thirdly, making the process knowledge explicit is beneficial for training process workers. Fourthly, explicit process knowledge enables offering a more uniform service, as the knowledge workers can compare and discuss the decision logic with each other in detail, which limits the amount of divergence (Shostack 1987). Lastly, processes can be optimized more efficiently and effectively because they capture the reality more closely (Wastell et al. 1994).

In this paper, the *Design Science Research* methodology (Hevner et al. 2004; Peffers et al. 2007) is followed in order to design a mixed-perspective process modeling language, called *DeciClare*, that can capture information related to all four perspectives (i.e., functional, control-flow, data and resource) when modeling loosely framed KiPs, for example, in healthcare and incident management. The language was built on a declarative foundation, because of its innate support for modeling flexible process flows. DeciClare integrates several existing extensions for declarative process modeling languages as well as new solutions for yet unaddressed shortcomings (i.e., general branching of constraints, exception handling, authorization constraints and pure time and data constraints) in order to support the strongly linked perspectives of such processes. Because the different perspectives had to be integrated into one approach, DeciClare can be categorized as a mixed-perspective modeling language (this classification is further explained in section 2.2).

The paper is structured as follows. In section 2.2, the related work is discussed, as well as the distinction between mixed-perspective, multi-perspective and hybrid languages. The applied research methodology is presented in section 2.3. In section 2.4, the problem stated in the section 2.1 is translated into a set of requirements that must be met in order to adequately model loosely framed KiPs. Section 2.5 then examines each perspective in isolation. Existing modeling languages and their extensions are shown to fulfill some of these requirements, however, additional solutions are also proposed for the requirements that are not yet met. These separate solutions are then integrated into one metamodel in section 2.6 for the new mixed-perspective language fulfilling all the requirements from section 2.4. Next, in section 2.7, the new language is evaluated by way of

expert validation based on a realistic KiP. Finally, the paper is concluded in section 2.9 and future work opportunities are described in section 2.10.

2.2. RELATED WORK

As stated in section 2.1, the languages that currently target loosely framed process (e.g., Declare and DCR graphs) do not sufficiently address all of its needs. Recently, some steps have been taken to add the support for different perspectives to Declare (Pesic 2008). However, most of the extensions have been presented as separate languages. The Declare⁺⁺ language is the only extension to combine multiple aspects of the functional/control-flow perspectives with a data perspective, but it does not fully support complex decisions (i.e., no disjunctions) and lacks a resource perspective. Compared to the solution proposed in this paper, Declare⁺⁺ is also less expressive when it comes to time concepts. Other declarative languages have recently also put more emphasis on the different perspectives of a process. Slaats et al. (2013) added a data extension to DCR graphs and Guard-Stage-Milestone language (Hull et al. 2010) was introduced, which incorporates a data-centric workflow model with declarative elements for modeling life cycles of business artifacts. These languages and language extensions are, however, not enough to model some of the more complex decisions found in KiPs, because they only allow simple decision logic (i.e., no disjunctions and decisions based on only one or at most just a few data values). Their focus is still very much on the process flow itself, and less on the decision processes that govern them. As stated by Di Ciccio et al. (2016), the integration of the existing extensions is considered an avenue for future work, and to our knowledge had not yet been addressed in a satisfactory manner. In this paper, the integration of the different perspectives has been performed, and additionally, new elements have been added as to increase the expressive power of the resulting language.

2.3. RESEARCH METHODOLOGY

This paper uses the Design Science Research methodology (Hevner et al. 2004; Peffers et al. 2007) to create the metamodel (i.e., the design artifact) of a process modeling language for loosely framed KiPs. We first defined the general requirements for the design artifact (section 2.4) based on literature (i.e., Req 1 and 2) and the lessons learned from the previous iteration of this design cycle (i.e., Req 3 and 4). These general requirements were subsequently translated to sets of subrequirements that define specific language constructs that needed to be supported by the design artifact. To satisfy the requirements, we identified all existing languages and language extensions from literature that support the corresponding language perspectives and analyzed which of the corresponding subrequirements could be satisfied with the ideas from the existing languages (section 2.5). For the subrequirements that were not yet satisfied, we then defined custom solutions.

As evaluation (section 2.7), we focused on the effectiveness of the proposed design artifact to solve the research gap that was identified in section 2.1 using the Conceptual Modeling Quality Framework (Nelson et al. 2012). For this we used the arm fracture case², which was first introduced in van der Aalst et al. (2009) and was extended in Mertens et al. (2015) through interviews with a practicing surgeon. In this paper, an expert panel of four medical practitioners with a real-life connection to the case (i.e., purposive sampling) was asked to evaluate the design artifact based on the case: an experienced general surgeon, an experienced orthopedic surgeon, an orthopedic surgeon in training (intern) and an experienced emergency nurse. By using practitioners that take on different roles in the process, we tried to get different viewpoints on the same case. The surgeons could validate/provide the medical diagnosis and treatment knowledge, the nurse could validate/provide the more practical knowledge on what happens starting from when the patient enters the emergency room and the intern the could validate/provide the academic knowledge as he was taught during his

² Note that the arm fracture case was only used during the evaluation phase. It was only during the writing of the paper that we decided to introduce the case earlier in the paper, so we could use it to illustrate the subrequirements and provide examples throughout the paper.

education. This should result in a more complete picture, with the maximum of variance in the type of knowledge that would need to be represented.

Design Science research is an iterative process (Hevner et al. 2004) and this paper presents our second iteration of the subject. This paper should not be considered an extension of the previous iteration (Mertens et al. 2015), but it rather restarts from scratch while incorporating the lessons learned from the previous iteration. Firstly, this paper abandons the ambition to propose both an abstract and a concrete syntax for the language in one paper. As stated by Moody (2009), both require a different approach and evaluation (i.e., semantic versus cognitive). Consequently, it is better to split them in two separate research steps so that each can be performed in greater depth and with a separate evaluation. This paper presents the first part of the research project: the development of an abstract syntax for a modeling language that targets loosely framed KiPs. An implementation of this abstract syntax in the form of an Ecore model is the resulting design science artifact. Secondly, this paper starts from a better theoretical foundation in the form of specific requirements for the language based on the needs of the targeted processes. This allowed for a more thorough analysis of the existing solutions (i.e., mostly extensions of declarative process modeling languages) as well as to identify remaining shortcomings. The resulting requirements thus served as a blueprint for designing the envisioned language. Lastly, an evaluation of the effectiveness of the proposed language by way of interviews with domain experts was also added.

2.4. LANGUAGE REQUIREMENTS

The *main goal* of this paper is the development of a process modeling language that allows for the functional, control-flow, resource and data perspectives of loosely framed KiPs to be modeled in an integrated manner. These perspectives should not be treated in isolation, because the different perspectives are naturally intertwined. For instance, the required execution of an activity (i.e., functional/control-flow perspectives) could tie up certain resources thereby impacting the general resource availabilities (i.e., resource perspective), while the activity itself could use and produce

certain data elements (i.e., data perspective). Alternatively, the unavailability of a resource (i.e., resource perspective) can result in the direct unavailability of an activity that requires the resource for its execution (i.e., functional/control-flow perspectives) and trigger decisions on how to respond to this unavailability (i.e., data perspective). Lastly, changing a data value (i.e., data perspective) could result in different activities needing to be executed (i.e., functional/control-flow perspectives) and adjusted resources availabilities (i.e., resource perspective). These domino effects between the perspectives are a common feature in many processes, but the relative prominence of all four of the perspectives in loosely framed KiPs (see section 2.1) increases the need to capture these effects in the process models. Therefore, the developed process modeling language will have to be of a mixed-perspective nature (see section 2.2) in order to integrate the different perspectives into one combined modeling technique.

The main goal can be translated into *five subgoals*: four subgoals expressing the need to support each of the perspectives and one additional subgoal for the integration of the different perspectives. These general subgoals can be operationalized to specific requirements for the modeling language by taking into account the properties and needs of loosely framed KiPs. There is one less requirement than there are subgoals, because the first requirement applies for both the functional and control-flow perspectives of the process. In the remainder of this section, these four requirements are presented and explained.

Req 1: The language must support ‘flexibility by design’

The first requirement relates to the *functional* and *control-flow* perspectives. Loosely framed processes allow many different arrangements of their activities, leading to a large variety of process variations being valid (Di Ciccio, Marrella, et al. 2015; Rovani et al. 2015). For example, take the diagnosis of a patient in an oncology department. The doctor can do any number of tests selected from a battery of possible tests to check whether or not the patient has cancer, and if so, determine the type, the size, the aggressiveness and the stage of the cancer. There are many different sequences

in which these tests can be done, but some sequences can or cannot occur depending on certain prerequisite conditions specific to each process instance. To model such highly-dynamic, human-centric, knowledge-intensive and non-standardized processes, a language is needed that supports the flexibility to model these variations and defers the choice for a certain variation to the run-time execution (Goedertier et al. 2015). This type of flexibility is called *flexibility by design* (Schonenberg et al. 2008). Flexibility by design thus implies that many alternative process variations are designed into the process model implicitly, allowing the choice of a suitable variation to be made at run-time for each individual process instance.

Req 2: The language must offer support for modeling complex decisions

The ‘knowledge’ in KiPs alludes to the implicit or explicit knowledge needed by the process workers to execute the process. Let us retake the example from the oncology department from the previous paragraph and suppose that the doctor has determined that the patient has a brain stem tumor. There are several ways for the process to proceed and, obviously, the process variation to follow is not chosen at random. To learn the type of cancer cells one would normally perform a biopsy, but, because of the location of the tumor, patient safety regulations state that this goes out the window. The doctor would have to resort to using an MRI, CT and/or some other imaging test to proceed. This is an example of a rather simple rule using process relevant data, in this case data generated during one or more previous activities, followed by a decision on how to proceed. KiPs are governed by many rules and/or decisions processes (Borrego and Barba 2014; Di Ciccio, Marrella, et al. 2015) and insight into these decisions is at least as important as what activities are available for execution, so support for decision modeling should be included in modeling language proposed in this paper. Because, at its core, decision modeling is nothing more than linking data values or combinations of data values to outcomes, this comes down to incorporating a *data* perspective into the language. The knowledge of a process is represented by the combination of specific data values and outcomes used in its decision-making. Typically, decisions that govern the

process execution of KiPs have a high perceived complexity (Di Ciccio, Marrella, et al. 2015). Some contributing factors to the decision complexity are the number of data values and data value aggregations used, the number of data sources and the number of decision conditions as well as their interrelations. The data perspective of the language thus needs to offer support for modeling complex decisions that guide process execution.

Req 3: The language needs to allow reasoning about resources

Resources play a vital role in loosely framed KiPs. More often than not, the execution of certain activities from the functional and control-flow perspectives depends on the availability of certain resources and those resources have certain properties expressed in data values (e.g., availability, speed, cost, quality, etc.) that influence the decisions in the data perspective of the process model. So, the modeling language needs to incorporate some form of *resource* perspective to help paint a complete picture of the process and its resources. For example, resources need to be taken into account to make the models executable, to allow a thorough analysis of the process, to offer meaningful guidance to process workers and to facilitate simulation and optimization techniques.

Req 4: The language must define a clear link between the different perspectives while carefully considering the domain appropriateness

Finally, the last requirement relates the previous three requirements. As stated at the start of this section, the four perspectives are closely linked. These relationships are complementary in most cases, however, overlaps exist. The functional/control-flow and data perspectives have an innate overlap in expressive power. For example, stating that activity B must be executed after activity A can, in theory, be represented as a part of either perspective. In the control-flow perspective it would result in a temporal relation between the two activities, while in the data perspective it would be a decision with the execution of activity A (= a fact, hence data) as condition and the execution of activity B as an outcome. This is an example of construct redundancy (Wand et al. 1999), signaling that there exist multiple ways to represent the same ontological concept in a language. According to

(Guizzardi 2013) the truthfulness of a modeling language with respect to the domain depends on the degree of isomorphism between the metamodel of the language and the domain conceptualization, also referred to as the *domain appropriateness*. Construct redundancy, excess and overload are all properties that serve as telltale signs of a misalignment of the domain appropriateness of a language. As a consequence, the way in which the four perspectives are linked and complement each other should be carefully considered.

2.5. DESIGNING THE PERSPECTIVES

In this section a standalone metamodel for each of the perspectives, required by *requirements 1, 2 and 3* from the previous section, will be developed. In section 2.6, we integrate these metamodels to also fulfil requirement 4. The concepts used in this section are defined on a rather abstract level to allow them to apply to a broad range of cases, which can make the link with real-life needs unclear. Therefore, a case description of realistic KiP will first be introduced to serve as a source of examples to demonstrate the practical relevance of the concepts used. The same case will also be used in section 2.7 to evaluate the proposed language.

2.5.1. The arm fracture case

As a practical example of a KiP, the realistic healthcare process of diagnosing and treating patients with suspected arm fractures created in Mertens et al. (2015) will be used (see Appendix A for a full description). This is not an isolated process in real-life, but rather a part of the general process spanning the emergency and orthopedic departments of a hospital. For this reason, we will use the term ‘realistic’ instead of ‘real-life’ to indicate the difference between the artificially reduced scope of the case compared to the even more complex real-life case. The scope of arm fracture case is limited to patients with one or more suspected fractures in his or her fingers, hands, wrists, forearms, upper arms, shoulders, and/or collarbones. The activities related to the admission, diagnosis and treatment phases are included (see Table 1).

In the remainder of this section, examples from this case will be used where possible. Just a few types of modeling concepts do not occur in this case. So, for these concepts examples from other healthcare processes will be used.

Admission	Register patient, Unregister patient
Diagnosis	Take medical history of patient, Clinical examination of patient, Doctor consultation, Take X-ray, Take CT scan, Apply intra-compartmental pressure monitor
Treatment	Prescribe NSAID painkillers, Prescribe SAID painkillers, Prescribe anticoagulants, Prescribe stomach protecting drug, Prescribe rest (=no treatment), Apply ice, Apply cast, Remove cast, Apply splint, Apply sling, Apply fixation, Remove fixation, Apply bandage, Apply figure of eight bandage, Perform surgery, Let patient rest, Stay in patient room

Table 1. The activities used in the arm fracture case

2.5.2. Declare as a foundation

Support for the flexibility by design principle (*requirement 1*) in the functional and control-flow perspectives can be achieved in multiple ways. In BPM there are two major modeling approaches: *imperative* and *declarative* process modeling (Fahland et al. 2009; Goedertier et al. 2015). Process modeling languages like BPMN (Object Management Group 2013) and Petri nets (Murata 1989; Petri 1962) are examples of imperative modeling languages (also known as procedural languages). These techniques can be described, in essence, as inside-out approaches to modeling because they describe exactly what the allowed behavior during process execution is (i.e., they provide an enumeration of all valid process variations) (van der Aalst et al. 2009). Consequently, imperative process modeling languages are well suited to model tightly framed processes as for such processes only a limited number of process variations need to be modeled. Declarative process modeling languages like Declare (Pesic 2008) and DCR graphs (Mukkamala 2012) take the opposite viewpoint with an outside-in approach. Declarative languages describe the rules and constraints by which the process must abide (Goedertier et al. 2015). The unwanted process variations are prohibited based on these rules and constraints, while all other variations are allowed without explicitly having to enumerate all valid variations (van der Aalst et al. 2009).

Although imperative process modeling languages can also support the flexibility by design principle, declarative process modeling languages are better suited for modeling loosely framed

processes as they do not require enumerating the potentially large number of valid process variations (Rovani et al. 2015). Generally, given the large number of allowed process variations in loosely framed processes, less effort will be required to describe the properties that make variations valid or invalid compared to enumerating all the valid process variations explicitly like imperative languages (van der Aalst 2013; Goedertier et al. 2015).

The Declare modeling language will be used as the foundation of our language, since it is currently one of the most popular and comprehensive declarative process modeling languages and it has many available extensions. Mulyar et al. (2007) has also shown that this language enables the representation of typical clinical guidelines, which are part of many loosely framed KiPs in healthcare. Declare is based on Linear Temporal Logic (LTL) (van der Aalst et al. 2009; Pesic 2008) and allows for process models to be defined in terms of activities and constraints spanning these activities. The process environment is specified in terms of what is necessary and what is not allowed (i.e., rules expressing the modal verb ‘must’ and ‘must not’), restricting the possible process executions. Contrary to other declarative languages like DCR Graphs and BPCN (Lu et al. 2009), Declare also supports guidelines (referred to as optional constraints). Such constraints offer guidance (i.e., rules expressing the modal verbs ‘should’ and ‘ought to’), while their soft character ensures flexibility is maintained (i.e., it is not necessary to enforce them) (Goedertier et al. 2015).

There are three main groups of Declare constraint templates (i.e., templates that can be used to create actual constraints in Declare models) defined in the description of the Declare language (Pesic 2008):

- Existence constraints: unary constraints predicating the allowed or mandatory amount of executions of an activity (e.g., `existence(3, A)` stating that activity A must be executed at least 3 times).

- Choice constraints: n-ary constraints expressing a choice between activities from a given set (e.g., `choice(2, [A, B, C])` stating that at least 2 distinct activities among A, B and C must be executed).
- Relation constraints: binary constraints enforcing a relation between two activities (e.g., `response(A, B)` stating that if A is executed, then B must be executed at any time after A). Also negative versions of these relation constraints exist (e.g., `responded_absence(A, B)` stating that if A is executed, then B can never be executed).

In the arm fracture case, for example, the general flow of the process states that a doctor can choose from several activities to reach a diagnosis and a wide range of possible treatments activities. However, an X-ray should be taken in almost every case (we will ignore the exception of life-threatening cases for now), immediately followed by a consultation with the patient to discuss the results. Imperative modeling language would require the explicit enumeration of every valid sequence of diagnosis and treatment steps. For example, a variation needs to be defined for when the doctor decides on a treatment consisting of an X-ray, a consultation, a prescription for painkillers and the application of a sling. But it should also be possible to give the patient the painkillers and/or sling before taking the X-ray. So even for this simple case, which does not even take into account the any prior steps, it will result in six variations depending on the order of these steps. Declare allows this to be modeled with just two constraints: *existence(1, Take X-ray)* and *chain_response(Take X-ray, Doctor consultation)*. All the variations that do not violate these constraints are implicitly allowed.

We created a metamodel of the Declare language (Fig. 5), because a complete metamodel of Declare is not available in literature. This metamodel is based on a partial metamodel of the Declare language (Bernardi et al. 2012) and the implicit metamodel of the concrete syntax proposed for the language (Pesic 2008) and the corresponding modeling tool³. The constraint templates themselves

³ <http://www.win.tue.nl/declare/>

Fig. 5. The metamodel of the Declare language

have been omitted from the metamodel to maintain a manageable overview and will be presented later on. Having an explicit metamodel for a modeling language is important as it represents the core concepts of the language and their relationships. This prevents possible ambiguity, which can make a language more difficult to use. It is also a necessary step to be able to correctly integrate new elements into a language and for any kind of tool support to be developed.

The metamodel in Fig. 5 captures the Declare language as it has been used in most research, with the existence and choice templates largely corresponding to the functional perspective and the relation templates to the control-flow perspective. However, there is one change that could already be made. This entails using events in addition to activities, similar to how this was proposed in (Montali 2010). Pesic (2008) defines the Declare constraint templates using activities as parameters. However, in the partial metamodel proposed in Bernardi et al. (2012) constraints are actually already linked with events as opposed to activities. This ambiguity is probably due to the use of events in the underlying LTL definitions for each of the templates in (Pesic 2008) and could have been avoided by providing an explicit metamodel of the language. The effect of allowing the templates to use event parameters is that it allows for much more fine-grained control when it comes to the expressive power of the constraints. Activities have a life cycle consisting of a start, end and cancellation event, mirroring the LTL definitions in Pesic (2008). For example, stating that a physical examination cannot be performed before the doctor or nurse has already started asking the

patient questions about his/her medical history, can now be expressed with a precedence constraint: *precedence(start(Take medical history of patient), start(Clinical examination of patient))*. This still leaves the option to perform these activities simultaneously or sequentially. In comparison, the original precedence template is always assumed to relate the end event of the preceding activity to the start event of the subsequent activity (i.e., *precedence(end(Take medical history of patient), start(Clinical examination of patient))*). Other declarative languages like DCR graphs also work on the level of events for similar reasons (Slaats et al. 2013).

During the application of Declare in specific process contexts, several shortcomings of the original scope of the language have been noted (e.g., resources, time, etc.). In response to these shortcomings, many standalone extensions have been proposed for Declare in the past few years. These extensions will be used as inspiration or starting point for the proposed solutions for the other requirements.

2.5.3. Extending the functional and control-flow perspectives

With Declare as the foundation to build on, the match between its expressive power concerning the functional and control-flow perspectives and the needs of loosely framed KiPs can be analyzed. We specified a set of seven rule patterns that typically occur in this type of process based on the overview of rule patterns in Caron et al. (2013) (i.e., A, B and C) and additional insights obtained from studying several loosely-framed processes in healthcare organizations (i.e., D, E, F and G). Each pattern can entail multiple constraints that differ in effect but are all related to a same underlying concept. These rule patterns, accompanied by an example of such a constraint in healthcare diagnosis and treatment processes, are presented in Table 2.

The available Declare templates can immediately fulfil several rule patterns. The existence templates of Declare can enforce the cardinalities of rule pattern A (for the example from Table 2: *existence(1, start(Register patient))* and *absence(1, start(Register patient))*). Rule pattern B can be fulfilled with the choice and responded existence templates A (for the example from Table 2:

choice(1, start(Perform surgery), start(Apply cast), start(Apply fixation), start(Apply bandage), start(Prescribe painkillers))). Constraints belonging to rule patterns *C* and *D* can be expressed by using the relation templates (for the example from Table 2: *response(end(Perform surgery), start(Prescribe anticoagulants))* and *not_chain_succession(end(Prescribe NSAID), start(Prescribe SAID))*). And lastly, Declare also supports optional constraints, which can be used to model the guidelines of rule pattern *G* (for the example from Table 2: *optional response(end(Perform surgery), start(Doctor consultation))*). However, support for rule patterns *E* and *F* is lacking, and thus, needs to be addressed.

	Rule pattern	Example
<i>A</i>	<i>Cardinalities</i>	A patient must be registered exactly once
<i>B</i>	<i>Coexistence</i>	At least two of the following treatments will be administered: surgery, a cast, a fixation, bandages and painkillers
<i>C</i>	<i>Temporal relationships</i>	If a patient has had surgery, then he will receive anticoagulants afterwards
<i>D</i>	<i>Negative relationships</i>	NSAID and SAID painkillers should not be administered simultaneously
<i>E</i>	<i>n-ary relationships</i>	Taking a CT or X-ray scan must be followed by a consultation with a doctor
<i>F</i>	<i>Exception handling</i>	If a patient has an adverse reaction to the anesthesia causing the doctors to abandon the rest of the surgery, this is followed by a consultation to determine what needs to happen next
<i>G</i>	<i>Guidelines</i>	Directly after surgery, the responsible doctor should visit the patient to explain what was done and what is next (i.e., a consultation)

Table 2. Seven typical rule patterns for flexible and KiPs

Before addressing the unsupported rule patterns, we will make a small change to the choice templates that were used to support rule pattern *B*. The exclusive choice template is replaced by the *AtMostChoice* template (the normal choice template will also be renamed to *AtLeastChoice*, see Table 7). The exclusive choice template states that exactly the given number of activities from the given set must be executed. Keeping in mind that the normal choice template expresses the same thing but with the ‘exactly’ replaced by ‘at least’, it seems strange to ignore that the ‘exactly’ can also be replaced by a template expressing ‘at most’. Similar to the other templates of the existence type, the ‘exactly’ is merely a combination of an ‘at least’ and an ‘at most’ constraint with the same

bound and activities. Therefore, adding the *AtMostChoice* template results in an increase of the expressive power and symmetry of the constraint templates.

Rule pattern *E* is the first unsupported rule pattern that will be discussed, as this entails a fundamental change to the Declare definitions. It conveys the need to express the constraints with more than the predefined number of connected elements. In the original definitions of the Declare constraint templates, constraints are connected to one (i.e., existence, with the exception of the choice templates) or two activities (i.e., relation) each. By *branching* the constraint templates, each connected element is replaced by a non-empty set of connected elements (e.g., *response([A and B], [C or D or E])*). Pesic (2008) already described this as a possible extension of Declare, while later work elaborated further on the subject of branching. Montali (2010) provided a definition for branching relation templates, while the TBDeclare extension (Di Ciccio et al. 2016) focusses on branching of the consequence side (sometimes referred to as the target side) of relation templates. The broadest definition of branching will be used, and thus, allow branching for all constraint templates. Although the definitions in the above-mentioned extensions always use an inclusive disjunctive interpretation of the branched parameters, DeciClare will sometimes also allow conjunctions and represent them as logical expressions. The introduction of branching impacts the templates in different ways:

- The original definitions of the choice templates are, in a sense, already branched according to the interpretation of a disjunction. This will be expanded with conjunctions. For example, the choice template (later referred to as the *AtLeastChoice* template) can be used to specify that at least one out of the three following options must be executed: *w* or *x* or (*y* and *z*). This constraint would allow event traces like *[w]*, *[x]*, *[y, z]*, *[w, x, y]* ..., but not *[y]*, *[z]*, *[y, y]*...
- The absence template (later referred to as the *AtMost* template) now supports both the inclusive disjunction and the logical conjunction for its logical expressions. For example, consider an event trace of *[x, x, x, z, y, y]*. It is correct to state that *x* or *y* occur at most 5 times (i.e., *AtMost([x or y], 5)*), corresponding to the sum of the occurrences of *x* and the

occurrences of y being at most 5. On the other hand, it is correct to state that x *and* y occur at most 2 times (i.e., $AtMost([x \text{ and } y], 2)$), corresponding to the minimum number of occurrences of x and y separately being at most 2. This is because at most 2 (x, y) -pairs can be found in the trace.

- For the existence template ($AtLeast$) similar logic applies for the disjunction. However, the conjunction is not supported because, for example, requiring at least a certain number of occurrences of (x, y) -pairs is equivalent to requiring x and y to separately occur at least that many times (i.e., two separate $AtLeast$ -constraints). To represent this in one statement could be useful for a concrete syntax, but in the metamodel of the language this will be avoided because this is syntactic sugar (see section 2.6.3).
- The relation templates, as presented in Montali (2010), support the inclusive disjunction. But this can also be further extended to include the expression of logical conjunctions. For example, consider a response constraint $Response([x \text{ or } [y \text{ and } z]], [w])$. This expresses that when event x has been executed or both events y and z have been executed, event w eventually needs to be executed. The example used in Table 2 can now be modeled as $Response([end(Take \text{ CT scan}) \text{ or } end(Take \text{ X-ray})], [start(Doctor \text{ consultation})])$.

Rule pattern F corresponds to exception handling constraints. It can be supported by introducing a *failure event*. Similar to the cancellation event, it marks an alternative ending of the execution of the corresponding activity, although this time not in a user-driven fashion (e.g., due to a change of heart by the patient). The constraint templates can now be used to specify what must happen when failure occurs. However, some combinations make little or no sense (e.g., the end event of activity A must be responded by the failure of activity B), while others are just hard to enforce (e.g., the failure of activity A can occur no more than once). Our suggestion would be to use the event only in combination with the responded existence, response and precedence templates (including their variations) with the additional limitation that it cannot be included in the condition parameter connected to the precedence templates, nor in the consequence parameter connected to the other

templates. For instance, the failure event can be used in a ChainResponse constraint to express that if an MRI scan fails because the patient had unexpected metal items in his/her body (e.g., hip replacement), then immediate surgery must be performed to repair the damage done by the MRI magnet. The example used in Table 2 can now be modeled as *ChainResponse([failure(Perform surgery)], [start(Doctor consultation)])*.

To allow for a shorter notation of the constraints, we will introduce a default event interpretation of an activity in a constraint. The start event of the activity will be the default interpretation for the existence and choice templates when only the activity name is used in the template. For example, *AtLeast([start(Register patient)], 1)* is the same as *AtLeast([Register patient], 1)*. For the relation templates, we will use the end event as the default interpretation for the condition side and the start event for the consequence side. For example, *Response([end(Perform surgery)], [start(Apply cast)])* is the same as *Response([Perform surgery], [Apply cast])*. Note that this means that the default interpretation of the previous example does not state that the cast must be successfully applied, just that the activity of applying the cast must be started. This leaves room for exception handling behavior to be specified, as the successful execution of an activity cannot be guaranteed.

2.5.4. Adding a notion of time-awareness to the constraint templates

Dadam et al. (2000) stated that the minimal and maximal time before, after and between activities is an important aspect in many loosely framed processes. The rule patterns presented in Table 2 do not deal with time as an explicit concept, as they largely coincide with expressive power of the original Declare language. Caron et al. (2013) also proposed two more rule patterns that deal with time as a concept (*T1* and *T2* in Table 3). Additionally, Lanz et al. (2014) have suggested a set of time patterns for supporting *time-awareness* in process-aware information systems based on empirical evidence from case studies (*T3-T7* in Table 3). Some of the time patterns suggested in Lanz et al. (2014) have been omitted, because they do not apply in the context of loosely-framed KiPs or are redundant as a result of other patterns presented throughout this paper.

Declare supports an implicit idea of time (e.g., sequencing), but it lacks the support for *metric time* as is needed to satisfy the rule patterns in Table 3. However, the idea of supporting metric time in Declare is not new. Barba et al. (2012), Montali (2010) and Westergaard and Maggi (2012) have all identified a need for constraint templates using time units. To solve this, a reformulation of Declare, called Timed Declare, was proposed in *Metric Temporal Logic* (MTL) that adds the option to express time units (Westergaard and Maggi 2012). As a result, new versions of all Declare templates were defined. For example, the timed response template can express that a doctor consultation needs to take place at some point in time, t_1 , after the taking an X-ray at t_0 , with $t_1 \in [t_0 + a, t_0 + b]$. The inclusion or exclusion of the boundaries of the interval can be specified in the template using the parameters a and b which are expressed in time units (b is allowed to be ∞). When a and b respectively equal 0 and ∞ , the formula collapses to the original definition of the response constraints of Declare. These MTL-definitions of the templates will be used instead of the original LTL-definitions as a starting point for our DeciClare language, because this reformulation increases the expressive power of Declare with a notion of time-awareness.

	Time pattern	Example
<i>T1</i>	<i>Time-oriented existence</i>	The triage must be started within 5 minutes of the process start.
<i>T2</i>	<i>Time-oriented relationships</i>	After surgery for a displaced or unstable fracture of the leg, a cast is applied. This cast cannot be removed within the first 10 weeks after surgery.
<i>T3</i>	<i>Lag between two activities</i>	The time between on the one side removing a cast and having done an X-ray and on the other side starting a doctor consultation, should be at most 15 minutes.
<i>T4</i>	<i>Durations</i>	The triage should not exceed 10 minutes.
<i>T5</i>	<i>Lag between arbitrary events</i>	The time between the start of the triage and the start of an examination by a doctor should not exceed 3 minutes for patients with a high-risk level.
<i>T6</i>	<i>Schedule restricted elements</i>	Lab tests can be requested on Mondays between 8 am and 5 pm.
<i>T7</i>	<i>Cyclic elements</i>	Patients with medium risk level should be checked every hour.

Table 3. Seven time patterns to complement the control-flow perspectives

The time parameters of the Timed Declare constraint templates add a notion of time to the constraint templates that is sufficient to satisfy time patterns *T1* (for the example from Table 3:

AtLeast([*Triage*], 1, 0, 5minutes)) and *T2* (for the example from Table 3: *NotResponse*([*Perform surgery*], [*Remove cast*], 0, 10weeks)). However, the other time patterns are not supported yet. *T3*, *T4* and *T5* can be treated jointly as the language already works on the granularity level of events. The time between two activities (*T3*) is defined as the time between the end event of the first activity and the start event of the second activity. *T5* generalizes this to the time between any two events of the process. This means that it also covers the duration of an activity (*T4*), because the duration is equal to the time between the start and end event of the activity. As a result, two new time constraints will be defined to add support for these three patterns: *AtLeastLag* and *AtMostLag*. These constraint templates have two logical expressions using events (inclusive disjunction and conjunction) as input parameters and express that the time between the satisfaction of the first expression and the second expression can be no less or no more, respectively, than a given limit expressed in time units (for the example from Table 3: *AtMostLag*([*end(Remove cast) and end(Take X-ray)*], [*start(Doctor consultation)*]), 15minutes)). The sixth time pattern (*T6*) makes it possible to express that certain activities are only available for execution according to a certain schedule. This can be generalized to also include expressing that certain activities are unavailable for execution according to a schedule (e.g., maintenance). For this, another two constraints are added: *EventAvailabilitySchedule* and *EventUnavailabilitySchedule*. Their parameters are a logical expression using events (inclusive disjunction) and a schedule. A schedule is a set of statements that take on the following form ‘(mandatory) every [*periodic point in time*], (optional) between [*two smaller periodic points in time*]’. The example from Table 3 would now be modeled as *EventAvailabilitySchedule*([*Lab tests*], [*every Monday between 8 am and 5 pm*])). The last time pattern (*T7*) enables the use of cyclic elements in the process. Where the Timed Declare definitions can be used to specify that an activity or event must be executed a certain number of times within a certain time interval, *T7* adds to option to specify the time lag between the different executions. This can also be modeled with the first two new time constraint templates (i.e., *AtLeastLag* and *AtMostLag*), and so, requires no additional changes.

Taking all of the changes from this subsection and the previous one into account produces the extended Timed and Branched Declare metamodel in Fig. 6. This represents the functional and control-flow perspectives in response to *requirement 1*. Note that the left side of the metamodel corresponds with the functional perspective and the right side with the control-flow perspective of DeciClare.

2.5.5. Supporting the data perspective

Fig. 6. The metamodel of the functional (left) and control-flow (right) perspectives

Until recently, no attempt had been made to support the data perspective in declarative languages (Debois et al. 2015). The languages lacked the expressive power to adequately model the (tacit) knowledge that governs the execution of loosely framed and KiPs (Lenz et al. 2012; Mertens et al. 2015). Recent publications (Maggi, Dumas, et al. 2013; Montali et al. 2013; Slaats et al. 2013) have proposed extensions that add some data-awareness to the main declarative languages, including Declare. Even though this is a good start, none of the proposed extensions contain all of the elements needed to offer a complete solution for *requirement 2*. The extensions allow for data to be linked to constraints of the control-flow perspective (cf. section 2.6.1), but they do not sufficiently specify the data perspective itself (e.g., Which data attributes are there? When are they assigned/changed? etc.).

The decision modeling standard from the OMG group (Object Management Group 2016), *Decision Model and Notation (DMN)*, served as inspiration during the formulation of a metamodel for the Declare data perspective. The primary goal of DMN is to provide a common notation for decision logic that can be understood by business users, business analysts and technical developers. This new standard is a response to the rising awareness that decisions can be just as important as, if not more important in some cases than, the process control-flow. By separating decisions from the process control-flow, the decision rules can be explicitly specified (Van Der Aa et al. 2015; Taylor et al. 2013). This means that the decision logic can be communicated, reused, adjusted to evolve within an ever-changing environment and used as justification for the choices being made.

DMN provides the constructs to model a complex decision in terms of its information requirements and the decision rules specifying the actual decision logic. For this purpose, a DMN decision model consists of two parts: a decision requirement graph and the decision logic. The former describes where the required information is coming from and can be depicted in one or more decision requirement diagrams. The latter describes the logic behind the decision, which is depicted in decision tables (Codasyl 1982). Decision tables were chosen as the only representation of decision logic currently supported by DMN, because they are very popular and easy to understand. The current minimal scope of DMN will be expanded to include other representations (e.g., decision trees) and allow for references to other types of models (e.g., SBVR) in the future as the standard evolves (Vanthienen 2014). The standard also describes its link to the popular imperative process modeling language BPMN explicitly through the Decision Task, but it positions itself as a separate perspective usable in other contexts as well (Object Management Group 2016). However, it is unclear how it can be used in combination with other languages, including declarative process modeling languages.

For the purpose of this paper, the focus will only be on the underlying concepts used by DMN to model decisions. The same concepts will be integrated in our metamodel, as to keep the option open to also use the same visual notations in future research. The upper half of a decision table specifies

the possible combinations of conditions, while the bottom half contains the actions to be taken (i.e., outcomes). This allows for complex decision logic to be modeled in one decision table or be spread over multiple tables. Fig. 8 shows the metamodel for decision tables. A metamodel of the decision requirement graphs, which can be used to specify where the data values in the decision tables come from as well as the relations between decision tables, has been omitted as these graphs are non-essential for representing the actual decision logic and do not require integration into the other perspectives.

Fig. 8. The metamodel of the DMN language for specifying decision logic (based on Figure 51 from (Object Management Group 2016))

The decision rules in Fig. 8 use unary tests as input entries. In turn, these tests use the values of data attributes to return a Boolean answer (e.g., patient gender='female'). Adding the concepts of a simple data model to Fig. 8 results in Fig. 7. A data record (e.g., patient file), possibly containing multiple sub records (e.g., medical history), consists of a set of data attributes (e.g., patient allergies), which each have one or more values connected to them (e.g., allergies={ 'penicillin', 'peanuts' }).

Fig. 7. The metamodel of the proposed data extension

The unary tests are now Boolean statements using these values (e.g., ‘the patient has an allergy to penicillin?’).

Finally, a last remark on how the decisions themselves are to be modeled in DeciClare. The principles of imperative and declarative modeling can, in a sense, also be applied to decision logic. Imperative decision logic could correspond to decisions that have explicit outcomes for all possible combinations of the conditions (i.e., satisfy completeness property). Contrarily, a declarative description of decision logic could correspond to decisions that do not fulfill this property. These decisions only describe the outcomes for a subset of all the possible combinations of conditions, leaving the rest unspecified. For example, a decision based on one Boolean data attribute is specified completely (i.e., imperatively) if separate actions are defined for when the data value equals ‘True’ and when the data value equals ‘False’. A declarative decision based on one Boolean data attribute specifies an action for one of the data values and leaves the other one unspecified. This follows the over- and underspecification tendencies of, respectively, the imperative and declarative way of thinking. As DeciClare takes on a declarative viewpoint, it follows that the specification of decision logic will also use this viewpoint.

2.5.6. Supporting the resource perspective

The resource patterns defined in Wil M.P. van der Aalst (2011a), Russell et al. (2004) and Russell et al. (2005) can be used as inspiration for adding an resource perspective to the language (*requirement 3*). The patterns show typical use of resources during the execution of workflow processes. These patterns are also used to evaluate the resource perspective of workflow languages and systems based on their expressive power (Mulyar 2005; Wohed et al. 2009). Workflow processes are classified as tightly framed processes, so they do not fully correspond to our target processes. However, the patterns can also be used to identify general aspects related to resource use and availability that are applicable to loosely framed KiPs. Table 4 presents those resource patterns that are not linked to other perspectives, in section 2.6.2 the other resource patterns will be presented. Based on the existing patterns, two additional patterns have been identified (denoted by an asterisk

in Table 4), which are important for our context. The new patterns are related to resource availability. In order for the allocation of resources to happen correctly, information is needed about the availability of single resources and the amount of resources that can fulfill a certain role.

	Resource pattern	Example
R1	<i>Direct availability*</i>	The hospital has two operating rooms (no.1 and no.2) and operating room no.1 is only available on Wednesdays between 8 am and 11 am.
R2	<i>Role-based availability*</i>	The hospital has 20 available receptionists.
R3	<i>Simultaneous execution</i>	A recovery room can simultaneously handle up to 10 patients.

Table 4. Three resource patterns relevant to flexible and KiPs

To improve the practical usability of the original Declare language, a resource extension was already proposed, called ConDec-R (Jiménez-Ramírez and Barba 2013). It enables reasoning about resources in Declare:

- Defining resources and the number of available instances of each resource type
- For each activity, linking the resource(s) required for its execution
- For each activity, adding estimates of its duration, cost or any other property

As there is again no metamodel publicly available, a metamodel was created based on the proposed semantics of Jiménez-Ramírez and Barba (2013) (Fig. 9). The ConDec-R extension does not clearly define what types of resources are included. Based on the example usage of the language in Jiménez-Ramírez and Barba (2013), resource reasoning capabilities were added with non-human resources in mind. However, the definition of the resource-element in the metamodel can be

Fig. 9. The metamodel of the resource extension of the ConDec-R language

expanded to also include human resources. Note that the metamodel already links Resource to Activity, and thus provides a link between the resource and functional/control-flow perspectives. This part of the metamodel will be discussed in section 2.6.2.

In terms of the resource patterns of Table 4, this metamodel directly satisfies pattern *R2*. The example from Table 4 can be modeled as *ResourceAvailability(Receptionist, 20)*. However, that metamodel cannot express, for example, that the sum of available doctors and nurses is five (independent of the actual distribution). By introducing branched resource constraint templates, similar to the control-flow constraint templates, this can be modeled as *ResourceAvailability([Doctor and Nurse], 5)*. Additionally, some extra flexibility can be added by specifying a lower and upper bound similar to the existence templates, instead of specifying the exact number of available instances of a resource. Branched resource constraint templates *AtLeastAvailable* and *AtMostAvailable* have been added, which have a logical expression using resources (inclusive disjunction), a lower or upper bound and two time bounds as parameters. For example, *AtLeastAvailable([Doctor and Nurse], 5, 0, ∞)* expresses that the sum of available doctors and nurses at any time during the process must be at least five. Although it is not directly required by the resources patterns, resource versions of the *ResourceAvailabilitySchedule* and *ResourceUnavailabilitySchedule* templates (*T6*) have also been added. The reasons for some activities to be (un-)available according to a certain schedule typically directly originates from the availability of a certain required resource according to the same or similar schedule. A typical example of this would be lab tests. Activities like performing blood and DNA testing rely on the availability of the lab for their execution. By adding these templates, the models can represent the reason for this (un-)availability more precisely (i.e., Perform DNA test requires a lab and *ResourceAvailabilitySchedule([Lab], [every Tuesday between 11 am and 4 pm and every Friday between 9 am and 6 pm])*)).

Support for the direct availability of resources (*R1*), for example stating that operating room no.1 is only available on Tuesdays between 8 and 11 a.m., can be added by allowing a resource to be

either a *direct resource* (e.g., dr. Gregory House or operating room no.1) or a *resource role* (e.g., orthopedic surgeon or operating room). A direct resource then of course can be part of a resource role (e.g., operating rooms no.1 and no.2 can satisfy the role of operating room: *Operating room(operating room no.1, operating room no.2)*). By allowing the resources roles to be further specialized, even more elaborate reasoning about resources is made possible (e.g., a doctor can have a specialization of neurosurgeon).

The last unsupported resource pattern deals with resources that can handle multiple resource allocations simultaneously (*R3*). This can be implemented by introducing one last constraint template: *SimultaneousCapacity*. It has a resource and an integer as input, combined with the standard two time parameters. The example from Table 4 can now be expressed as follows: *SimultaneousCapacity([Recovery room], 10, 0, ∞)*.

Integrating the solutions for the metamodel of the resource perspective, results in the metamodel presented in Fig. 10.

Fig. 10. The metamodel of the proposed resource extension

2.6. INTEGRATING THE PERSPECTIVES

The previous section proposed extensions for Declare resulting in three standalone metamodels, enabling the functional/control-flow, data and resource perspectives of a process to be represented.

These metamodels are merely partial solutions to *requirements 1, 2 and 3*, because these perspectives are heavily linked to one another. So, the full expressive power of each perspective can only be achieved by considering its relations to the other perspectives. In this section, we will start from the metamodel of the functional and control-flow perspectives (Fig. 6) and step-by-step define the links with the data (Fig. 7) and resource (Fig. 10) perspectives. These links will open the door to expand the expressive power of the latter two perspectives even further. At the end, everything is integrated into one consistent metamodel that defines the new declarative process modeling language for loosely framed KiPs.

2.6.1. Integrating the data perspective

	Data pattern	Example
<i>D1</i>	<i>Data-driven existence</i>	If the patient has an open, displaced or unstable fracture, surgery will be performed at least once.
<i>D2</i>	<i>Data-driven relationships</i>	If the patient is a minor, then consent is required from the legal guardian before surgery can be performed.
<i>D3</i>	<i>Insert data</i>	If the patient does not have a patient file yet for this hospital, a new one needs to be created during the registration of a patient.
<i>D4</i>	<i>Read data</i>	The doctor will need to examine the patient file during a consultation.
<i>D5</i>	<i>Update data</i>	If the patient reports information that does not correspond to his/her patient file (e.g., a different allergy) when taking his/her medical history, the patient file will be updated.
<i>D6</i>	<i>Delete data</i>	If a previously place internal fixation has been removed from the patient's arm, the data attribute in the patient's file stating that he/she has an internal fixation will be deleted.

Table 5. Six data patterns for KiPs

The rule patterns proposed by Caron et al. (2013), which were already used in sections 2.5.3 and 2.5.4, also include two rule patterns (*D1* and *D2* in Table 5) that deal directly with the link between the data perspective (Fig. 7) the one hand and the functional/control-flow perspectives (Fig. 6) on the other hand. Yet, similar data patterns can be defined by taking inspiration from database applications, which are typically used to store and access the data in practical applications. Database query languages, like the popular Structured Query Language (SQL) (ISO/IEC 2011), have defined

a set of operations that can be performed on data: INSERT, SELECT, UPDATE and DELETE. This corresponds to how data records and their data attributes can be created, read, modified and removed during the execution of activities of a process. Based on these data operations from SQL, four additional data patterns have been formulated (*D3*, *D4*, *D5* and *D6* in Table 5).

Data patterns *D1* and *D2* essentially require the option for all constraint templates supported by the functional and control-flow perspectives to link them with data. As a result, constraints would not need to apply for all process executions, as is customary in most declarative languages, but can apply for a mere subset of executions. For example, the two examples from Table 5 can be modeled as *if*('Open fracture = true' or 'Displaced fracture = true' or 'Unstable fracture = true') then *AtLeast*([Perform surgery], 1, 0, ∞) and *if*('Patient is a minor = true') then *Precedence*([Get consent from legal guardian], [Perform surgery], 0, ∞). To express that a constraint does apply to all process executions, it just needs to be linked with a trivial activation decision (i.e., always true). This will be represented in DeciClare by a constraint without the 'if'-part. For the constraint templates with a condition side (sometimes referred to as the source side), the activation decision on its own is not enough to activate the constraint (e.g., RespondedExistence, Response, Precedence...). Only the fulfilment of both the activation decision and condition of the constraint are enough for an activation. For example, consider a response constraint stating that if the patient has a displaced fracture, then surgery needs to be followed by applying a cast to the leg (i.e., *if*('Displaced fracture = true') then *Response*([Perform surgery], [Apply cast], 0, ∞)). For an activation of this constraint, the patient needs to have a displaced fracture *and* the doctors must have performed surgery. If one of these two conditions is not met, the constraint cannot be activated. An inactive constraint holds trivially (i.e., cannot be violated) and is commonly referred to as a vacuous satisfaction (Burattin et al. 2012). Hence, if the patient does not have a displaced fracture, the previous example constraint is not active and therefore vacuously satisfied.

Linking the functional, control-flow and data perspectives this way means that a constraint is always connected to an activation decision and the outcome of the decision is the activation of the

constraint itself, while in the process making the `LiteralExpression` concept obsolete. This approach combines the power of the decision conditions to specify complex decisions based on data with the expressive power of the supported functional and control-flow perspectives as outcomes. This solution is similar to the decision-dependent constraints from Mertens et al. (2015) and the first-order logic expressions from Maggi, Dumas, et al. (2013), albeit here at a more general level. The decision-dependent constraints and first-order logic expressions can be seen as instantiations of the `UnaryTest` concept from Fig. 7. Linking the perspectives this way, has some consequences that need to be discussed in more detail and that sometimes also influence the design decisions made in previous section.

First, an important question that arises: does a constraint that is activated after some activities have been performed cover only the activities after its activation, or also the activities that preceded its activation? The latter interpretation, that constraints apply retroactively, might be useful when the models are used for conformance checking of past execution, but can make enforcing them practically impossible when using the model to execute the process. For example, a constraint stating that a certain activity can only be executed once, can be activated after it has already been executed multiple times. Requiring the executing process instances to satisfy constraints up to and after the constraint activation seriously limits the expressive power and does not correspond to how decisions are taken in real life processes. So, the applicability of an activated constraint will be defined as only the activities after the constraint activation.

A change to the definitions of the timed existence constraint templates is now required to maintain the consistency, because these templates specify that the time parameters (e.g., *a* and *b*) are relative to the process instance start, and thus, could have a retroactive effect. In order to prevent this, these parameters will be defined as relative to point in time when the constraint is activated. So for example, a timed ‘`AtLeast`’-constraint could specify that a clinical examination must take place within the first hour after the patient was marked as ‘yellow’ during the triage (i.e., *if(‘Triage color = yellow’) then AtLeast([Clinical examination of patient], 1, 0, 1hour)*), independent of the fact that

the patient might have arrived 20 minutes before he/she was triaged. The parameters can still express time relative to the process instance start by having a trivial activation decision or an activation decision that evaluates as true at the start of the targeted process instances.

Second, having a decision to activate a constraint is one thing, but in some cases, it can be necessary to deactivate an activated constraint later on. Consider for example a constraint expressing that patients with a certain type of cancer need to receive at least six chemotherapy treatments. There are situations in which this constraint needs to be deactivated: if the patient reconsiders his consent after less than six treatments, if the patient experiences one or more complications after a couple of treatments and even the worst-case scenario when the patient deceases before at least six treatments have been administered. As these reasons could not be foreseen when formulating the activation decision for the constraint, a way to deactivate constraints is needed. So, every constraint will be given both an activation and deactivation decision. Of course, it is again allowed to be a trivial decision (i.e., always false) to represent a constraint that can never be deactivated. A constraint is not allowed to be violated as long it is activated but does not always need to be fulfilled completely like as illustrated in the following example. Suppose the patient wants to stop the treatment after just two of the six treatments, then the constraint is deactivated even though the minimum of six treatments prescribed by the constraint was never satisfied. Of course, if an activated constraint would put a maximum on the number of treatments, this maximum is not allowed to be broken in the timeframe between the activation and deactivation of the constraint, although it is allowed after the deactivation. A consequence of having both an activation and deactivation decision is that is possible that a constraint is activated and deactivated multiple times during one process instance. For example, take a constraint expressing that if a patient has fluid in his/her lungs, then this needs to be drained within 3 minutes unless the fluid has already been drained or dissolved by itself (i.e., *if('Fluid in lungs = true') then AtLeast([Drain fluid], 1, 0, 3minutes) unless if('Fluid in lungs = false')*). Fluid is found the first time, the constraint is activated and the fluid will be drained. Afterwards, the constraint will be deactivated as the fluid has been drained. However, suppose the

fluid immediately comes back. This activates the constraint again, leading to another deactivation after a second drainage.

Finally, a last unaddressed issue concerns the overlap in expressive power between the functional/control-flow and data perspectives (see requirement 4 in section 2.4). The main reason for this overlap is that the decisions can specify the same conditions as the conditions of the relation templates. For example, consider the response constraint stating that taking an X-ray must be followed by a consultation (i.e., $Response([Take\ X-ray], [Doctor\ consultation], 0, \infty)$). This can also be expressed with an existence one constraint combined with an activation decision depending on the execution of the ‘Take X-ray’ activity (i.e., *if* (‘Take X-ray has been executed = true’) *then* $AtLeast([Doctor\ consultation], 1), 0, \infty$). This issue is resolved by not allowing decisions to use statements about the execution of events and activities. This eliminates the overlap, but at the same time reduces the expressive power of the resulting language. For instance, modeling a constraint like ‘if surgery has been executed, then taking an X-ray will have to be executed at least 2 times’ is no longer possible. To circumvent this side effect of the proposed solution, we will modify the relation templates to make such constraints possible once again. It comes down to allowing the relationship ends to be expressed in terms of existence templates. More specifically, the changes to the relation templates entail using branched existence ($AtLeast$), absence ($AtMost$), choice ($AtLeastChoice$) and $AtMostChoice$ (see section 2.6.3) templates to express both the condition and consequence sides. Like the branching principle discussed earlier, a logic expression connecting these templates with the logical conjunction and inclusive disjunction operations will be used as condition or consequence side. The previous example can now be modeled as follows: $Response(AtLeast([Perform\ surgery], 1), AtLeast([Take\ X-ray], 2), 0, \infty)$. As a side effect, the negative versions of the relation templates are no longer needed explicitly, since they can be expressed by using the normal templates with an absence 0 ($AtMost\ 0$) constraint as its consequence side. For example, consider a constraint stating that if an MRI has been taken, an X-ray cannot be taken afterwards. In Declare this would be written as $NotResponse([Take\ MRI], [Take\ X-ray])$, but

we could now write this using the normal response-template as *Response(AtLeast([Take MRI], 1), AtMost([Take X-ray], 0), 0, ∞)*. The end result of this set of changes to the relation templates can express everything we could before the changes but has eliminated the overlap and adds even more expressive power by making it possible to express relations between multiple executions of events.

Now that support for the first two data patterns has been added, the last four patterns (*D3*, *D4*, *D5* and *D6*) need to be addressed. For each of these data patterns two constraint templates can be defined to express whether this operation must occur or cannot occur. This results in the following constraint templates: RequiredInsertion/ProhibitedInsertion, RequiredRead/ProhibitedRead, RequiredUpdate/ProhibitedUpdate and RequiredDeletion/ProhibitedDeletion. The templates relate data structures to activities of the process. The parameters of these templates are branched in a similar manner as for the control-flow constraint templates: a logical expression using activities (inclusive disjunction) and a logical expression of data structures (inclusive disjunction). Additionally, each of these templates has time parameters and an activation and deactivation decision. The last four examples of Table 5 can now be modeled as follows:

- *D3: if('Patient has been to this hospital before=false') then RequiredInsertion([Register patient], [Patient file], 0, ∞)*
- *D4: RequiredRead([Doctor consultation], [Patient file], 0, ∞)*
- *D5: optional RequiredUpdate([Take medical history of patient], [Patient file], 0, ∞)*
- *D6: RequiredDeletion([Remove fixation], [Patient file/fixations], 0, ∞)*

2.6.2. Integrating the resource perspective

The resource patterns defined by Wil M.P. van der Aalst (2011a), Russell et al. (2004) and Russell et al. (2005) were used in section 2.5.6 to define a resource perspective for DeciClare. However, several patterns could not yet be discussed as they require links to the other perspectives. Table 6 presents the remaining resource patterns relevant to loosely framed and KiPs. Again, an additional resource pattern was added (denoted by an asterisk). This pattern is a variation on the existing

authorization pattern. The original pattern covers the authorization of a role to execute a certain task, while the added authorization pattern covers the authorization of roles to take certain decisions. As KiPs typically involve multiple process workers, establishing these authorization rules can be crucial in order to successfully execute the process.

	Resource pattern	Example
R4	<i>Direct allocation</i>	Brain surgery is performed in operation room no.1.
R5	<i>Role-based allocation</i>	Applying a cast requires at least one doctor or one nurse.
R6	<i>Activity-based authorization</i>	A surgeon is authorized to perform surgery.
R7	<i>Decision-based authorization*</i>	A doctor is authorized to decide whether or not surgery to be performed.
R8	<i>Automatic execution (no human resources)</i>	Monitoring the heartrate of the patient is executed by the heartrate monitor machine.
R9	<i>Separation of duties</i>	For surgery, one person cannot take on both the surgeon and anesthesiologist roles.

Table 6. Six additional resource patterns linking the resource perspective to the other perspectives

The metamodel of the ConDec-R language (Fig. 9) directly satisfies resource pattern *R5*, so a similar solution will now be added to the metamodel of the resource perspective of DeciClare (Fig. 10). Analogous to how support was added to express resource availabilities, new constraint templates are added that use branching and upper/lower bounds to express resource allocations: *AtLeastUsage* and *AtMostUsage*. They have five parameters: a logical expression using activities (inclusive disjunction), a logical expression using resources (inclusive disjunction), a lower or upper bound, and two time parameters. They can express that a certain activity needs at least or at most a certain number of resources (possibly, but not necessarily, of the same type). For example, it can state that applying a cast requires at least one doctor or one nurse: *AtLeastUsage([Apply cast], [Doctor or Nurse], 1, 0, ∞)*.

No further changes are required to support the direct allocation of resources (*R4*), as this is done analogous to the direct availability of resources (*R1*). For example, stating that all brain surgeries

must be performed in operating room no.1 can be expressed as *AtLeastUsage([Perform brain surgery], ["operating room no. 1"], 1, 0, ∞)*.

The authorization patterns (e.g., *R6*, *R7* and *R8*) are also not yet supported by the metamodel. To implement these patterns two additional pairs of constraint templates are added: *(Not)DecisionAuthorization* and *(Not)ActivityAuthorization*. These templates have as input on the one hand a logical expression using resources (inclusive disjunction) and on the other hand, respectively, a constraint or an activity. They express that the connected resources have authorization to take the decision connected to a certain constraint or execute a certain activity. The examples for *R6*, *R7* and *R8* from Table 6 can now be modeled as follows: *ActivityAuthorization([Orthopedic surgeon], [Perform surgery])*, *DecisionAuthorization([Doctor], [if('Displaced fracture?' = true' or 'Open fracture?' = true') then AtLeast([Perform surgery], 1, 0, ∞)])* and *ActivityAuthorization([Heart rate monitor], [Monitor heart rate])*. The last template can also be used to, for example, express that an expert system has authorization to diagnose a patient autonomously (i.e., *ActivityAuthorization([Expert system], [Diagnose patient])*).

The final resource pattern is the one that expresses the principle of separation of duties (*R9*). To support this pattern, two more resource constraint templates are introduced. The *ResourceEquality* and *ResourceInequality* constraint templates are connected to two or more pairs of logical expressions, each consisting of one resource expression (inclusive disjunction) and one activity expression (inclusive disjunction). This allows for expressing that resources from certain activities must be the same or must be different, respectively, as the resources in some other activities. Consider the example from Table 6 for *R9*, this can now be modeled as *ResourceInequality([Orthopedic surgeon or Anesthesiologist], [Perform surgery])*. An example of a resource equality on the other hand, would be to express that the surgeon that performed the surgery needs to be the same doctor as the one that does the follow-up examination of the patient (i.e., *ResourceEquality([Orthopedic surgeon], [Perform surgery or Doctor consultation])*).

2.6.3. DeciClare

The metamodel of the DeciClare modeling language, which fully integrates the functional/control-flow, data and resource perspectives presented earlier, is presented in Fig. 11. Some constraint templates without additional relations have been omitted to improve readability. An Ecore implementation of the complete metamodel has been made available⁴.

Fig. 11. The metamodel of the DeciClare language (blue = functional, purple=control-flow, orange = data and green = resource)

As summary of the solutions for each of the perspectives, the resulting constraint templates are presented in Table 7, Table 8 and Table 9. For Table 7, the second column contains the constraint templates from the most expressive Declare extension Declare⁺⁺. In Table 8 and Table 9 this column was omitted, as both perspectives are not supported by Declare⁺⁺. The last column of each table relates the constraint template to the corresponding patterns from Table 2, Table 3, Table 4, Table

⁴ DeciClare_ecore.zip available at dx.doi.org/10.1016/j.eswa.2017.06.024

5 and Table 6. The parameters X, I, C and B define the logical operators that can be used in the parameter expressions of the template as a result of the branching concept. The used letter refers to the option to use the exclusive disjunction (X), inclusive disjunction (I), conjunction (C) or both the inclusive disjunction and conjunction (B). The numerical bound enforced by some templates is expressed by the parameter n. The time interval linked to the Timed Declare definitions is represented by parameters a and b. The set of schedule expressions is denoted by S ($T6$). To save space, the activation and deactivation decisions for each constraint template and their optional versions have been omitted. The optional versions of each of the templates (rule pattern G) are of course still supported in the same way as the respective mandatory versions. In addition to presenting the solutions for the functional and control-flow perspectives of DeciClare, Table 7 also shows the results of a systematic review based on the principle of domain appropriateness (i.e., *requirement 4*) of the existing constraint templates. Some of the templates from Declare have been renamed in order to clarify their meaning and to emphasize the symmetry of the templates, while others have been omitted because they are either redundant (i.e., exclusive allowance) or syntactic sugar. The latter means that they can be easily expressed in terms of the other templates. Syntactic sugar can be categorized as construct redundancy, and thus, needs to be avoided in order to meet *requirement 4*. It is something that could be reintroduced when defining a concrete syntax to improve usability, but it does not belong in the metamodel of the language.

Type	Declare ⁺⁺	DeciClare	Patterns
Existence templates	existence(n, Activity, a, b)	AtLeast(I, n, a, b)	A, E, T1
	absence(n, Activity, a, b)	AtMost(B, n, a, b)	A, E, T1
	exactly(n, Activity, a, b)	<i>(removed: syntactic sugar)</i>	-
	init(Activity)	<i>(removed: syntactic sugar)</i>	-
	last(Activity)	<i>(removed: syntactic sugar)</i>	-
	choice(n, I, a, b)	AtLeastChoice(B, n, a, b)	B, T1
	exclusive choice(n, I, a, b)	<i>(removed: syntactic sugar due inclusion of AtMostChoice)</i>	-
	-	AtMostChoice(B, n, a, b)	B, T1
	-	EventAvailabilitySchedule(I, S)	E, T6
	-	EventUnavailabilitySchedule(I, S)	E, T6
Relation templates	responded existence(I ₁ , I ₂)	RespondedPresence(B ₁ , B ₂)	B, D, E
	responded absence(I ₁ , I ₂)	<i>(removed: expressed as RespondedPresence with AtMost-consequence)</i>	-
	coexistence(I ₁ , I ₂ , a, b)	<i>(removed: syntactic sugar)</i>	-
	not coexistence(I ₁ , I ₂ , a, b)	<i>(removed: syntactic sugar)</i>	-
	response(I ₁ , I ₂ , a, b)	Response(B ₁ , B ₂ , a, b)	C, D, E, T2
	not response(I ₁ , I ₂ , a, b)	<i>(removed: expressed as Response with AtMost-consequence)</i>	-
	chain response(I ₁ , I ₂ , a, b)	ChainResponse(B ₁ , B ₂ , a, b)	C, D, E, T2
	not chain response(I ₁ , I ₂ , a, b)	<i>(removed: expressed as ChainResponse with AtMost-consequence)</i>	-
	alternate response(I ₁ , I ₂ , a, b)	AlternateResponse(B ₁ , B ₂ , a, b)	C, E, T2
	precedence(I ₁ , I ₂ , a, b)	Precedence(B ₁ , B ₂ , a, b)	C, D, E, T2
	not precedence(I ₁ , I ₂ , a, b)	<i>(removed: expressed as Precedence with AtMost-consequence)</i>	-
	chain precedence(I ₁ , I ₂ , a, b)	ChainPrecedence(B ₁ , B ₂ , a, b)	C, D, E, T2
	not chain precedence(I ₁ , I ₂ , a, b)	<i>(removed: expressed as ChainPrecedence with AtMost-consequence)</i>	-
	alternate precedence(I ₁ , I ₂ , a, b)	AlternatePrecedence(B ₁ , B ₂ , a, b)	C, E, T2
	succession(I ₁ , I ₂ , a, b)	<i>(removed: syntactic sugar)</i>	-
	not succession(I ₁ , I ₂ , a, b)	<i>(removed: syntactic sugar)</i>	-
	chain succession(I ₁ , I ₂ , a, b)	<i>(removed: syntactic sugar)</i>	-
	not chain succession(I ₁ , I ₂ , a, b)	<i>(removed: syntactic sugar)</i>	-
	alternate succession(I ₁ , I ₂ , a, b)	<i>(removed: syntactic sugar)</i>	-
	-	AtLeastLag(B ₁ , B ₂ , n)	E, T2/3/4/5/7
	-	AtMostLag(B ₁ , B ₂ , n)	E, T2/3/4/5/7

Table 7. The control-flow constraint templates of Declare⁺⁺ and DeciClare

Type	DeciClare	Patterns
<i>Data templates</i>	RequiredInsertion(I_{act}, I_{data}, a, b)	$D3$
	ProhibitedInsertion(I_{act}, I_{data}, a, b)	$D3$
	RequiredRead(I_{act}, I_{data}, a, b)	$D4$
	ProhibitedRead(I_{act}, I_{data}, a, b)	$D4$
	RequiredUpdate(I_{act}, I_{data}, a, b)	$D5$
	ProhibitedUpdate(I_{act}, I_{data}, a, b)	$D5$
	RequiredDeletion(I_{act}, I_{data}, a, b)	$D6$
	ProhibitedDeletion(I_{act}, I_{data}, a, b)	$D6$

Table 8. The data perspective constraint templates of DeciClare

Type	DeciClare	Patterns
<i>Resource templates</i>	AtLeastAvailable(I_{res}, n, a, b)	$R1, R2$
	AtMostAvailable(I_{res}, n, a, b)	$R1, R2$
	ResourceAvailabilitySchedule(I_{res}, S)	$T6$
	ResourceUnavailabilitySchedule(I_{res}, S)	$T6$
	SimultaneousCapacity(res, n)	$R3$
	AtLeastUsage(A, I_{res}, n, a, b)	$R4, R5$
	AtMostUsage(A, I_{res}, n, a, b)	$R4, R5$
	ActivityAuthorization(I_{res}, act)	$R6, R8$
	NoActivityAuthorization(I_{res}, act)	$R6, R8$
	DecisionAuthorization($I_{res}, constraint$)	$R7, R8$
	NoDecisionAuthorization($I_{res}, constraint$)	$R7, R8$
	ResourceEquality($[I_{res}, I_{act}]_{2..*}$)	$R9$
	ResourceInequality($[I_{res}, I_{act}]_{2..*}$)	$R9$

Table 9. The resource perspective constraint templates of DeciClare

2.7. EVALUATION

Hevner et al. (2004) and Peffers et al. (2012) present several types of evaluation that can be used to evaluate design science artifacts. A combination of expert evaluation and an illustrative scenario Peffers et al. (2012) has been used to evaluate the artifact of this paper, which is the abstract syntax of the DeciClare modeling language. More specifically, domain experts have evaluated the abstract syntax with the help of the realistic arm fracture case, presented in section 2.5.1, as application scenario. The goal of this evaluation was to validate the effectiveness of the abstract syntax of the

DeciClare language. The reasons for including certain modeling concepts in the abstract syntax have already been discussed in sections 2.4-2.6 and were each based on either literature or practical examples (i.e., informed arguments (Hevner et al. 2004)). In the first subsection, the set-up of the evaluation will be discussed. In the subsequent subsections the results of the evaluation concerning different quality types will be presented.

2.7.1. The evaluation set-up

The effectiveness of the design artifact, the metamodel of the DeciClare modeling language, can be assessed by applying the Conceptual Modeling Quality Framework (CMQF) (Nelson et al. 2012). This framework proposes several quality types based on what are called the quality cornerstones of conceptual modeling. The cornerstone that serves as the subject of our evaluation is the DeciClare metamodel, which in terms of the CMQF corresponds to a physical language that attempts to allow users to create a faithful representation of relevant concepts and phenomena in the physical domain, in this case loosely framed and KiPs.

In order to evaluate the DeciClare metamodel both directly and indirectly a realistic healthcare process of diagnosing and treating patients with suspected arm fractures was introduced. Because a concrete syntax of DeciClare was not yet available, the textual representations of the proposed metamodel, as used throughout this paper, were used to serve as a temporary implementation of a concrete syntax for the DeciClare language. To make it as easy as possible for the participants, the metamodel instantiations were subsequently translated to more understandable natural language (see Appendix B for the original DeciClare model). For example, the constraint stating that surgery must be performed within 1 hour if the symptoms for acute compartment syndrome have been fulfilled was originally modeled as *“AtLeast(Perform surgery, 1, 0, 1hour) activateIf[Has acute compartment syndrome? = true]] deactivateIf[False]”* and now becomes *“If('Has acute compartment syndrome? = true') then Perform surgery must be performed at least 1 time within at most 1 hour”*. The original model of the case was based on the available literature and contained 15 activities, 11 resources, 15 data attributes and 79 constraints (35 functional/control-flow, 37

resource, 7 data and no guideline constraints) using 10 different constraint templates of the 22 available across all perspectives (not counting the mandatory/optional, positive/negative and at most/at least versions separately).

Using both the metamodel and the instantiation of the model, the CMQF framework can be used to evaluate the effectiveness of the language with three different quality types: perceived semantic quality, pragmatic quality and language-domain appropriateness. The perceived semantic quality of the case model represents the extent to which the model accurately and completely captures the domain knowledge of the experts, within the scope of the modeling task at hand (Krogstie 2000). During the interviews issues with the semantic quality of the case model were treated as opportunities. They allow for discussions on the exact origin of the encountered issues and possible solutions. For instance, problems with the completeness of the language could be revealed, if it is not possible to formulate a satisfactory solution (i.e., missing concepts or relations in the DeciClare metamodel). The pragmatic quality on the other hand captures the extent to which the experts completely and accurately understand the parts of the case model that are relevant to their roles in the actual process. Basically, this gives an idea of the usefulness and value of the case model to the stakeholders of the process, and per extension of modeling with DeciClare in general.

Both perceived semantic quality and pragmatic quality indirectly evaluate the DeciClare metamodel by means of the developed model. The language-domain appropriateness quality type of the CMQF has as object of interest the metamodel of the language. It is the quality type related to the relationship between the physical language and the physical domain. However, an objective measurement of this relationship is impossible, because the universe of discourse related to the domain of loosely framed and KiPs is impossible to capture in its entirety. To circumvent this issue, a more subjective quality type can be used as approximation: the perceived language-domain appropriateness (Nelson et al. 2012). This quality type states that the metamodel, as understood by the users/modelers, must be appropriate to the understanding of the real-world universe of discourse by these users/modelers and for the ultimate use of the models created with the modeling language.

This can be assessed by domain experts, who possess the in-depth knowledge of loosely framed KiPs to assess the appropriateness of the DeciClare metamodel to allow creating faithful representations of such processes.

By way of several semi-structured interviews with domain experts, all three quality types were evaluated for the DeciClare modeling language (see Appendix C for the interview protocol). In the first part of the interview, the expert was introduced to the research project and the DeciClare modeling language. In the second part, a brief introduction to the arm fracture case and its scope was given, followed by some time to thoroughly read the case description. When the experts were fully aware of the scope of the case and sufficiently understood the modeling language, they were presented with the natural language model of the case. All the rules of the model were discussed one by one as part of the semantic quality evaluation. If a rule did not conform to reality, corrections were discussed. In the last part of the interview the pragmatic quality of the model was discussed followed by a general discussion about the perceived language-domain appropriateness of the modeling language. The arm fracture case takes place in the emergency and orthopedic departments of hospitals. Therefore, four medical professionals that fulfill different roles along these departments in three of the biggest hospitals in Ghent were interviewed as domain experts: a surgeon from the AZ Sint-Lucas hospital, an emergency nurse/process quality promotor from the Ghent University Hospital and both an orthopedic surgeon and an orthopedic surgeon in training (intern) from the AZ Maria Middelaers hospital in Ghent. The first three were very experienced and offered detailed insight into the process in practice, while the orthopedic surgeon in training was able to cast a different light on the process based on his training and his recent emergence into the complexity of real-life cases.

2.7.2. Evaluating the perceived semantic quality

In order to verify the semantic quality of the model the participants were instructed to validate the constraints of the model in a per-context (e.g., all relevant constraints when a patient with a finger fracture enters the emergency department) way on their correctness and completeness. This revealed

some interesting feedback on the model of the case: new activities (e.g., ‘Prescribe stomach protecting drug’), data values (e.g., ‘Displaced fracture’) and constraints were added, while others were modified and some even removed. The reasons ranged from a lack of detail in the original model concerning certain treatments and decisions to a sense that certain deadlines were difficult to enforce or merely hospital policies that differ from hospital to hospital. All of the suggested changes could be incorporated into the model with the available modeling constructs supplied by the DeciClare metamodel as to the eventual satisfaction of the experts. So, this did not expose any issues concerning the completeness of the language. After each interview the model was updated with additional insights, resulting in a final model containing 25 activities, 17 resources, 21 data attributes and 282 constraints (126 functional/control-flow, 115 resource, 7 data and 34 guideline constraints) using 16 different constraint templates (see Appendix D). Most of the DeciClare examples used throughout this were taken from this updated model of the arms fracture case.

2.7.3. Evaluating the pragmatic quality

The pragmatic quality of the case model assesses the extent to which the experts understood the model and were able to use it. To this purpose, the experts were asked to identify examples of conforming and non-conforming process instances based on the model. All of them were able to perform this task satisfactory. For the non-conforming examples, they were also able to pinpoint the violated constraint or set of constraints.

Furthermore, the experts were asked if they could use the model to support their daily job. The responses confirmed the usefulness of the model. They all agree with the statement made earlier in the paper, that it can be a valuable tool when training new personnel. The orthopedic surgeon in training also added that he had not encountered anything similar during his education and training. They were taught using all sorts of specific clinical pathways (Rotter et al. 2010), but never got a higher level view of the process. He agreed that a model like this could have the potential to significantly shorten the internship duration, as they would no longer be limited to gaining experience from shadowing cases of more experienced doctors. Also, the experts see it as a great

way to communicate changes to the process, be it due to policy changes, new diagnostic tests or new treatments. However, it was noted that the direct value of the model for a certain process actor somewhat diminishes as the experience level of the process actor rises. This was to be expected, as a rising experience level means that the knowledge made explicit by the model becomes more and more known. So less experienced process actors will be able to learn more from the model, than those with more experience. Yet, as noted by the emergency nurse, more experienced process actors can still get more insight into how other process actors, fulfilling different roles, make decisions. Her role in the process is to prepare the patient before seeing the doctor and provide support throughout. The model could help her identify and acquire the specific bits of information that the doctors will need in subsequent activities faster.

The experts were also asked how they currently use the information described in the model in their daily job. While they acknowledged the tacit nature of the knowledge by referring to its application as a sort of autopilot mode, they did say that they think more in terms of clinical pathways due to the nature of their training. The constraints expressed by the model refer to one or a few activities directly or to a relation between them, however, the experts would group them more according to certain clinical pathways (e.g., a Colles' fracture). An interesting feature to consider would be to allow filters on the model, showing only the constraints that apply for specified data values. This would allow the constraints for each clinical pathway to be presented separately, while still being able to connect them to the complete process. This is somewhat similar to how Maggi et al. (2011) filtered their Declare models based on previously executed activities to make them more understandable. Additionally, they indicated that in reality many decisions also come down to patient preferences. Typically, the patient is directly or indirectly responsible for much of the variability present in a healthcare process. For example, the constraints in the model do not account for patients that refuse treatment. Of course, the patient preferences can be represented by data values, but this would make the decision logic of the affected constraints more complex. The experts agreed that for communication purposes this would be unfavorable, but for analysis and training

purposes it could be very useful to know exactly which decisions need to account for patient preference and which do not.

2.7.4. Evaluating the language-domain appropriateness

In the last part of the interview, the DeciClare metamodel textual representations were discussed independently of the presented case. The experts agreed that the language is sufficiently expressive to also model the real underlying process without the limitations put on the scope (i.e., the complete emergency services process). They feel that the model would become a lot more complex, but that the added complexity would be in proportion to the added complexity of the real-world process. When considering applying the technique to other healthcare processes that fit the mold of flexible and KiPs, they were unable to think of examples (i.e., processes or types of constraints) that could not be expressed using the DeciClare language. The lack of identified construct deficits (Guizzardi 2013) at this point or during the discussion of the case model, allows us to conclude that there are no apparent issues with the completeness of the language. Also, no construct excesses (Guizzardi 2013) were uncovered, since real-life examples of each of the constraint templates could be given. Of course, this is because special attention was given to these properties during the design of the language.

One expert did, however, mention that after discussing some of the modeling elements from Declare that were left out due to being syntactic sugar, she would prefer to have these elements included to simplify the models. The example that we discussed was to include the ‘Init’ template, from the Declare language, to express that the patient registration must happen as the first activity of a process instance (*Init([Register patient])*). Instead, this was expressed by 24 precedence relations between the patient registration activity and every other activity (i.e., *Precedence([AtLeast([Register patient], 1)], [AtLeast([Take medical history of patient], 1)], 0, ∞)* and the same for the 23 other activities). As stated in section 2.6.3, we are certainly open to adding these types of modeling elements as part of the eventual concrete syntax of the language. Introducing the Init/Last templates and the ‘exactly’ versions of each of the AtLeast/AtMost templates would

have reduced the final case model from 282 to 218 constraints (78 functional/control-flow, 99 resource and 7 data and 34 guideline constraints). However, these templates were not included in the temporary concrete syntax used for the purpose of this evaluation, because it was a direct instantiation of the metamodel of DeciClare.

When considering the DeciClare language as a whole, the experts were most intrigued by how the different perspectives complemented each other. A good example of this from the case model were the rules that modeled what should happen when a patient has a confirmed displaced wrist or forearm fracture. The normal treatment involves surgery to reposition the broken bone (i.e., *if('Displaced fracture = true' or 'Open fracture = true') then AtLeast([Perform surgery], 1, 0, ∞)*), which requires at least a free operating room (i.e., *AtLeastUsage([Perform surgery], [operating room], 1)*), a surgeon (i.e., *AtLeastUsage([Perform surgery], [orthopedic surgeon], 1)*), an anesthesiologist (i.e., *AtLeastUsage([Perform surgery], [anesthesiologist], 1)*) and a surgical assistant or one or more OR or scrub nurses (i.e., *AtLeastUsage([Perform surgery], [surgical assistant or scrub nurse], 1)* and *AtLeastUsage([Perform surgery], [OR nurse], 1)*). But we also modeled the frequently occurring situation where the operating time (i.e., the time for which all the necessary resources need to be available) is not yet available (i.e., *if(['Displaced fracture? = true' and 'Operating time available? = false' and 'Confirmed Fracture? = Wrist'] or ['Displaced fracture = true' and 'Operating time available = false' and 'Confirmed Fracture = Forearm']) then Precedence([[AtLeast([Apply cast], 1) or AtLeast([Apply splint], 1)] and AtLeast([Stay in patient room], 1)], [AtLeast([Perform surgery], 1)], 0, ∞) unless if('Operating time available = true')*). Instead, the responsible doctor will order a temporary cast or splint to be applied (by himself or a nurse) to bridge the time until the operating time becomes available again. This situation could be modeled completely in DeciClare, because it supports and combines the necessary decisions, functional/control-flow restrictions and resource reasoning to express it. Therefore, the general conclusion of the experts was that the perceived model-domain appropriateness of DeciClare is

satisfactory and that it is sufficiently expressive to model loosely framed and KiPs with the needed level of detail.

2.8. POSITIONING OF DECICLARE IN THE RESEARCH DOMAIN

A summary of the expressive power of all languages and extensions used in this paper in relation to the requirements of section 2.4 and their operationalization by means of patterns in sections 2.5 and 2.6 is presented in Table 10. When a pattern is covered by the language this is denoted by ‘+’, when unaddressed by ‘-’ and when not applicable by ‘/’. In some cases, a certain pattern is somewhat addressed, but not to the full extent, which is denoted by ‘±’.

The language proposed in this paper, DeciClare, can be classified as a *mixed-perspective* process modeling language, as opposed to a *multi-perspective* process modeling language. The former combines perspectives by integrating the metamodels of the perspectives into one resulting metamodel. The latter combines perspectives by using them side by side, keeping the metamodels of these languages separated. Coloured Petri nets (Jensen 1997) is another example of a mixed-perspective language, as it uses one integrated metamodel to combine the control-flow perspective of regular Petri nets with a data perspective. The pockets of flexibility approaches of Sadiq et al. (2005) and De Smedt et al. (2016) are more examples of mixed-perspective approaches. The Declare extensions Declare⁺⁺, ConDec-R, data-aware Declare and Declare based on LTL-FO can also be classified as mixed-perspective modeling languages. The difference with DeciClare is that they cover at most three perspectives, where this paper covers the main four. On the other hand, Maggi, Slaats, et al. (2014) and Frank (2014) are examples of multi-perspective approaches combining perspectives related to representation (also called representation paradigms). For Maggi, Slaats, et al. (2014), either an imperative or declarative representation is chosen, depending on the characteristics of each sub-process in isolation. The two are never combined into one model. In Frank (2014), the metamodels of the different perspectives remain separated, with a meta-metamodel relating all the metamodels.

		<div>Declare (Pescic 2008)</div> <div>Timed Declare (Westergaard et al. 2012)</div> <div>TBDeclare (Di Ciccio et al. 2016)</div> <div>ConDec-R (Jiménez-Ramirez et al. 2013)</div> <div>Data-aware Declare (Borrego et al. 2014)</div> <div>Declare++ (Montali 2010; Montali et al. 2013)</div> <div>Declare based on LTL-FO (Maggi et al. 2013)</div> <div>DMN (Object Management Group 2016)</div> <div>Declare-R-DMN (Mertens et al. 2015)</div> <div>DeciClare</div>										
1	A	+	+	+	+	+	+	+	/	+	+	
	B	+	+	+	+	+	+	+	/	+	+	
	C	+	+	+	+	+	+	+	/	+	+	
	D	+	+	+	+	+	+	+	/	+	+	
	E	-	-	±	-	-	+	-	/	-	+	
	F	-	-	-	-	-	-	-	/	-	+	
	G	+	+	+	+	+	+	+	/	+	+	
	T1	-	+	-	-	-	-	-	/	-	+	
	T2	-	+	-	-	-	+	-	/	-	+	
	T3	-	-	-	-	-	-	-	/	-	+	
2	T4	-	-	-	-	-	-	-	/	-	+	
	T5	-	-	-	-	-	-	-	/	-	+	
	T6	-	-	-	-	-	-	-	/	-	+	
	T7	-	-	-	-	-	-	-	/	-	+	
		-	-	-	-	±	±	+	+	+	+	
	3	R1	-	-	-	-	-	-	-	/	-	+
		R2	-	-	-	+	-	-	-	/	+	+
		R3	-	-	-	-	-	-	-	/	-	+
	4	R4	-	-	-	-	-	-	-	/	-	+
		R5	-	-	-	+	-	-	-	/	+	+
R6		-	-	-	-	-	-	-	/	-	+	
R7		-	-	-	-	-	-	-	/	+	+	
R8		-	-	-	-	-	-	-	/	-	+	
R9		-	-	-	-	-	-	-	/	-	+	
D1		-	-	-	-	+	+	+	/	±	+	
D2		-	-	-	-	+	+	+	/	+	+	
D3		-	-	-	-	-	-	-	/	-	+	
D4		-	-	-	-	-	-	-	/	-	+	
D5		-	-	-	-	-	-	-	/	-	+	
D6		-	-	-	-	-	-	-	/	-	+	

Table 10. A comparison of the languages used in this paper based on the requirements from section 2.4 and their operationalization as patterns from section 2.5 and 2.6

The DeciClare language does not fall into the category of hybrid languages. Within business process modeling research, the category of ‘hybrid process modeling languages’ is used to signify the combination of imperative and declarative modeling languages into one approach (Goedertier et al. 2015; Madhavji and Schafer 1991; Maggi, Slaats, et al. 2014; Sadiq et al. 2005; De Smedt et al.

2016). The idea is that some of the information can be better expressed using an imperative language, while for other parts this is true when using a declarative language. So, they combine two (or more) paradigms, in the form of either a multi- or mixed-perspective language, to *represent* the information contained by the model. DeciClare uses only one paradigm for representation, the declarative paradigm, but instead focuses on combining different *content* related perspectives (like Coloured Petri nets).

2.9. CONCLUSION

In this paper the metamodel of the mixed-perspective declarative modeling language DeciClare has been presented. This metamodel is the final step in a journey that started with the formulation of requirements of a modeling language for loosely framed and KiPs. These requirements were translated into sets of smaller requirements (i.e., patterns) based on relevant literature, which represent the operationalization of the original requirements. Subsequently, the patterns were analyzed one by one and addressed. Although several existing languages already targeted this type of processes, none could offer the means to model all of its intricacies as specified by the requirements. The foundation for the DeciClare language was the popular declarative process modeling language Declare. Based on the requirements, an evaluation was performed to identify the aspects supported and unsupported by Declare. Because the Declare language received many standalone extensions since its conception, the relevant extensions were also evaluated in the same way. This helped reveal exactly those parts that could be reused and those that necessitated new solutions. After the development of separate solutions for the different perspectives in response to the requirements, an integration step was performed. The integration of the perspectives is an important step for the academic and practical applicability of declarative languages. The myriad of standalone extensions might each add support for some specific aspects, but an approach combining of all of these extensions is a vital step to capturing these processes completely in a process model.

An essential aspect of modeling KiPs is, of course, modeling the knowledge itself. This knowledge governs the process and all of its numerous variations. As it is typically of a tacit nature, it is paramount that a language to model these processes offers a way to explicitly represent the decision logic that expresses it. DeciClare links each declarative constraint with decision logic that can make the constraint active and that can deactivate an active constraint. Process models can therefore represent the process very precisely, as well as the knowledge governing it. This facilitates the sharing of knowledge with other knowledge workers and with other stakeholders of the process (e.g., the patient). Additionally, a more uniform and transparent service can be offered when the knowledge workers are more informed, and they have a way to communicate changes to the process at their disposal.

Along the way, this paper also makes several contributions to the research domain in the form of other metamodels. The first metamodel that was presented is the one of the Declare language (Fig. 5). Although a partial metamodel for Declare had been proposed before (Bernardi et al. 2012), it was not fully compatible with the original definitions of the language and its corresponding modeling tool. This metamodel was subsequently enriched with additional concepts and relations in response to the requirements. The resulting model (Fig. 6) presents a direct extension of the Declare language, since it originally only covered the functional and control-flow perspectives. When considering the addition of a data perspective, the concepts from DMN and a simple data representation were used to create a metamodel for the data side (Fig. 7). And finally, a metamodel of the resource perspective was presented in Fig. 10. This model was based on the proposed metamodel of ConDec-R extension (Fig. 9) for Declare, for which a metamodel was also lacking in literature.

The effectiveness of the abstract syntax of the DeciClare language was evaluated by way of semi-structured interviews with domain experts. An example of a loosely framed and KiP was used (i.e., a simplification of the emergency services process only considering patients with possible fractures in the arms) to make the discussions and questions more relatable for the experts. Based on a general

explanation of DeciClare, the case description and a textual DeciClare model of the case, the language-domain appropriateness of the language was assessed. The semantic and pragmatic quality of the case model were discussed, and afterwards a general discussion took place about the DeciClare language. This led to two main takeaways.

The first conclusion that can be drawn from the evaluation is that the DeciClare language appears to be sufficiently expressive to model loosely framed and KiPs. This will be further verified in the future by using DeciClare to model several more loosely framed KiPs (cf. chapter 4), preferably from different domains (e.g., legal profession). The original case model was enriched during each of the interviews, but nothing was encountered that could not be expressed in DeciClare. The domain experts saw no potential problems for the model of the case to be expanded to represent the unrestricted diagnosis and treatment process in the emergency service department, apart from the proportional increase in complexity. Additionally, they were unable to come up with constraints that could not be captured by a DeciClare model but might occur in different processes of this type.

The second conclusion is that the experts deemed modeling the processes with DeciClare useful for training and communication purposes. It can be a tool to make these processes, which are generally perceived as black boxes by management, more transparent to all stakeholders. It also allows for process actors to anticipate the needs of other process actors more correctly, because the models can offer insight into their role in the process and their corresponding needs. However, experience plays a vital role here. Inexperienced process actors will benefit more from the added insight, as more experienced actors already know much of the information relayed by the model.

In general, DeciClare offers modelers a new way to model loosely framed KiPs. Although it is a new language, it still has a familiar feel to it because existing modeling concepts were integrated whenever possible. Compared to alternative languages, the main advantage of DeciClare stems from its integration of the different perspectives of the process. Most of the languages targeting the same types of processes offer only one perspective to the process, resulting in important information getting lost during the modeling phase (as illustrated by Table 10). DeciClare allows each of the

essential perspectives of the processes to be represented, as well as the links between these perspectives. This is somewhat similar to what the BPMN/DMN language extension attempts to do for tightly framed processes. BPMN/DMN also combines the functional, control-flow, resource and data perspectives, but using the imperative modeling paradigm. On the flipside, approaches like DeciClare and BPMN/DMN introduce a lot of added complexity. It is a typical tradeoff between simplicity and expressive power. In case of DeciClare, we feel that the latter is justified because the targeted processes are inherently complex. The resulting models are still an abstraction of reality that offers a view of the process with reduced complexity, but they are also comprehensive enough to represent the most important intricacies of the process. Additionally, we envision most practical applications of these models (e.g., process mining, simulation, conformance checking and run-time support) to be supported by way of (semi-)automated techniques.

Finally, the DeciClare language was positioned in the business process modeling domain using several possible classifications. The first classification is according to the types of processes that the language targets, in this cases loosely framed KiPs. Of course, this was a design decision. The second classification positions the language according to what type of perspectives are combined in the language and how these different perspectives are combined. DeciClare can be classified as a *mixed-perspective* modeling language, because multiple *content* perspectives (i.e., functional, control-flow, data and resource) are integrated into one metamodel. The last classification is according to the usage scenarios supported by the language. At design-time it provides the means to convert tacit knowledge generally associated with loosely framed processes to explicit knowledge, allowing this knowledge encapsulated in the models to be communicated with others. The language makes it possible to explain and justify why things are done a certain way, by making the decision logic transparent for all stakeholders. In an execution context it facilitates conformance checking of past and present process executions.

2.10. LIMITATIONS AND FUTURE WORK

Firstly, the scope of this paper was limited to the abstract syntax of the modeling language. A visual syntax for a modeling language requires a different research approach (i.e., cognitive instead of semantic) and a different evaluation, and so we chose to split the two into separate research steps that each can be performed in greater depth. Therefore, the next step in this research cycle will be to develop a visual syntax for the DeciClare language.

A second limitation is intrinsic to declarative modeling, namely that the models tend to be complex and difficult to understand for human users. Of course, the choice for a good visual syntax will also play a role in this, but an extended metamodel will also see the complexity of possible models rise. There is some research being done on how to reduce the size of declarative models (Di Ciccio, Maggi, et al. 2015; Di Ciccio and Mecella 2015; Maggi, Bose, et al. 2013), which could help mitigate the problem. However, it is not yet clear if these techniques are still applicable, and if so, how they would be applied taking into account the introduction of activation decisions and deactivation decisions for constraints. Another possible way to reduce this issue came up during the evaluation, specifically the option to add data-filters to the visualization of a model. This makes it possible to view certain cases in isolation (e.g., clinical pathways), removing all unrelated or inactive constraints from the model.

A third limitation was already discussed in section 2.5.1, namely that the case used in this paper had some limitations on the scope. Instead of modeling the complete real-world process of the emergency and orthopedic departments of a hospital, we limited the scope to certain types of arm fractures. This was necessary to reduce the complexity process and its resulting DeciClare model to reasonable levels. As was noted by the domain experts during the evaluation, the added complexity would not have changed anything to the conclusions of this paper. In the next step of our research project, we are going to use DeciClare to model the whole real-life process in the emergency department of a big Belgian hospital. A final limitation relates to a design decision made in section

2.6.1. Although most of the language very much supports conformance checking, the choice to not have activated constraints apply retroactively can be seen as a small restriction on its conformance checking potential. However, these types of constraints would be very hard to enforce, as process actors would only know in hindsight whether some constraint should have been enforced. So, this concept is not very useful for any sort of active use of process models.

The last limitation relates to the scope of the decision modeling aspect of DeciClare. DeciClare focusses on the specific combinations of data values that lead to certain behavior. For example, if a patient has a suspected lower arm fracture then a scan needs to be made. Yet, these data values are just the consequences of an actual decision rationale that was applied. For example, a scan is needed to check if there is indeed a fracture and, if so, to determine the position of the fractured bone endings. The decision rationale is certainly more valuable knowledge. However, a huge time and/or effort investment of the knowledge workers is needed to make these decision rationales explicit. The only options to gather some data on decision rationales is through interviews or by asking the knowledge workers to think aloud during their daily tasks and recording this. The former is often inefficient and incomplete, while the latter is very invasive and difficult to use on a large scale. Nonetheless, including data on the decision rationales would certainly be beneficial when modeling loosely framed KiPs. In the future, an extension of DeciClare could be considered to include this, similar to how this is included in KiPN (Netto et al. 2019).

The future research plans start with the four limitations discussed above. Additionally, we will study how DeciClare can support simulation techniques and provide a foundation for offering guidance and recommendations to the process workers at run-time.

RESEARCH CONTRIBUTIONS OF CHAPTER 2

- The metamodel of the Declare process modeling language
- The metamodel of the ConDec-R extension of Declare
- The metamodel of the DeciClare modeling language for loosely frame KiPs
- A classification of (process) modeling languages: multi-perspective and mixed perspective

Chapter 3 describes EC4 (see section 1.4).

Reference:

Mertens, S., Gailly, F., Poels, G.: *Discovering healthcare processes using DeciClareMiner*. Heal. Syst. - Spec. Issue SIG-Health Des. Innov. Impact Healthc. IT. 7, 195–211 (2018).

3

Discovering Loosely Framed Knowledge-intensive Processes using DeciClareMiner

3.1. INTRODUCTION

Business process modeling has been successfully adopted in many different sectors. However, healthcare is one of the few sectors that have yet to fully embrace the process-driven approach (Lenz and Reichert 2007), even though some of the main concerns trending in eHealth are very similar to those of other sectors, namely, cost reduction, efficiency and patient orientation (Payton et al. 2011). The specific characteristics of healthcare processes and, more generally, of loosely framed KiPs are the primary cause (Di Ciccio, Marrella, et al. 2015). These processes typically have a set of possible activities that can be executed in many different ways, leading to situations where deviations and variations are the norm rather than the exception. The knowledge workers executing a process (e.g., doctors or other healthcare personnel) decide which activities to perform and when they are to be performed. To make these decisions, they often use what is called tacit knowledge: they have an implicit idea of the appropriate actions to perform when certain conditions apply (Lenz and Reichert 2007). This idea might be partially based on explicit knowledge (e.g., medical or hospital guidelines), but experience usually factors in heavily. Of course, it would be beneficial for an organization to not only transform these implicit ideas into explicit rules as much as possible to be able to provide a more consistent and transparent service but also allow making explicit changes to these rules to potentially improve the outcomes of a process.

Many currently available process modeling languages focus solely on the explicit sequencing of the activities (e.g., activity X is executed first, then Y, and finally Z) of these processes (i.e., imperative control flow). However, exposing the decisions that govern the sequencing of activities (e.g., activity Y must be executed because activity X has been executed and data attribute 'd' has value 19) can be just as important, if not more important, for loosely framed KiPs. As a result, these languages cannot be used to model the knowledge governing these processes. As part of our ongoing research project, we have developed a declarative process modeling language, called DeciClare, designed specifically for modeling these types of processes (Mertens et al. 2017). DeciClare uses decision logic as a complement to the declarative modeling concepts to provide the means to

describe processes from the data perspective. A new tier of applications is unlocked by modeling the processes with this higher level of detail. Consider the following example applications:

- The knowledge contained by models will make simulations much more realistic and flexible. For example, how would a department address an increase in a group of pathologies by 5%? Would this create new bottlenecks? Can these be eliminated by changing, if possible, some decision criteria?
- The decision logic of certain treatments in models can be compared to the available clinical guidelines. Do the doctors follow the guidelines? If not, what are the characteristic and decision criteria of the cases in which they diverge from the guidelines?
- The knowledge of the mined models can be used to bring interns up to speed quicker or even to educate future nurses/doctors as part of medical courses.

However, the manual creation of decision and process models of real-life loosely framed KiPs remains a problem. Parts of the knowledge that must be captured in these models is typically missing due to its tacit nature. This greatly hinders the adoption of many process modeling, analysis and redesign techniques for loosely framed KiPs. Therefore, the research question addressed by this paper is as follows: *“How can we automatically and efficiently discover DeciClare models, keeping in mind the computational challenge due to the vast amounts of data that need to be processed in realistic use cases?”*

The process mining domain provides ways to (semi-)automatically discover, monitor and improve real processes by extracting knowledge from the data logged by the IT systems of organizations during their day-to-day operations (van der Aalst et al. 2011). A simple example would be a production line company, where the employees execute their daily activities in a strict sequence while the IT system of the company logs the starts and end times of these activities. Process mining techniques can detect this pattern from the logs and automatically compose a process model that reflects the actual process correctly. The DeciClare language can represent the relevant

information of loosely framed KiPs, but a corresponding process mining technique for (semi-) automatically discovering these models is currently missing.

In this paper, we propose a process mining technique called DeciClareMiner that is specifically developed for loosely framed KiPs. This technique can mine a declarative process model and its corresponding decision logic (i.e., a DeciClare model). In essence, it attempts to offer a more complete picture of the (possibly tacit) knowledge found in loosely framed KiPs. This is achieved by mining the rules that have been applied, implicitly and/or explicitly, in past executions of the process, which are contained within a given event log. The two-phase solution strategy used by the miner to efficiently discover the models is the most important contribution of this paper. The strategy intelligently aligns an Apriori-inspired phase with a custom genetic algorithm to considerably reduce the vast search space.

The remainder of this paper is structured as follows. The next section gives an overview of the related literature. In section 3.3, a brief description of the used modeling language is provided. The research methodology that was applied is discussed in section 3.4. This is followed by a description of the proposed algorithm in section 3.5. In section 3.6, the algorithm is evaluated by applying it to a realistic healthcare case and comparing the results to a benchmark for different levels of complexity. Subsequently, the limitations of the presented algorithm are discussed in section 3.7. The last section concludes the paper and presents directions for future research.

3.2. RELATED WORK

The healthcare domain turned to process management due to the constant pressure to improve the quality of their services while at the same time reducing the associated costs (Anyanwu et al. 2003). Processes in healthcare can be classified as either organizational or medical treatment processes (Lenz and Reichert 2007). Medical treatment processes can be further categorized as either standard, routine or non-routine processes according to the level of variability and complexity (Lillrank and Liukko 2004). This paper is focused on routine and non-routine medical treatment processes. To

manage the variability and complexity of these processes, several approaches have been proposed. For example, Birkmeyer et al. (2001) advocated for the enforcement of hospital volume standards for high-risk procedures. The basic idea is that the large-scale repetition makes these hospitals more capable of managing the variability and complexity. However, Gawande stated that relying on process workers' memory alone is not enough when faced with the occasional extreme complexity and specialization of healthcare. He proposed making extensive use of checklists, similar to in the aviation sector, to reduce human mistakes due to forgetfulness (Gawande 2010). Another important way to manage medical treatment processes is the use of clinical pathways (De Bleser et al. 2006; Campbell et al. 1998; Rotter et al. 2010; Vanhaecht et al. 2006), which model the essential steps in the care of patients with specific clinical problems. Traditional business process management (Amorim Lopes et al. 2016; Faber et al. 2016; Helfert 2009; Hofmann and Perry 2010; Keel et al. 2017; Mazzocato et al. 2010) and modeling techniques (Aguilar et al. 2008; Braun et al. 2014; Noumeir 2006; Putman et al. 2015; Rojo et al. 2008; Svagård and Farshchian 2009) have also been applied in different healthcare scenarios.

In the process modeling literature, there are two modeling approaches (three when counting hybrid approaches): imperative and declarative modeling. The former approach explicitly models every possible sequence of activities in the process model, while the latter approach models only the rules that determine if a sequence of activities is (in)valid (van der Aalst et al. 2009). The latter is preferred when modeling non-standard healthcare processes, as they do not require enumerating the potentially large number of valid process variations (van der Aalst 2013; Goedertier et al. 2015; Rovani et al. 2015). Many process variations are modeled implicitly, allowing the choice of a suitable variation to be made at runtime for each individual process instance (i.e., episode of care).

Checklists are a form of declarative process model, as they state only that each activity on the list needs to be performed and not how or when. However, they offer limited support to process workers regarding in which situations a certain checklist should be used. Early declarative languages, such as Declare (Pesic 2008), tried to improve on this by allowing general relations

between possible activities to be described (e.g., if activity X is performed, then activity Y will need to be performed somewhere in the future). This allows for a checklist, of which activities need to be performed, could be performed and should not be performed, at any point during the execution of a process to be created dynamically. DeciClare (Mertens et al. 2017) takes this a step further by allowing the models to specify the actual decision logic, if available, regarding the time at which certain activities must be performed, could be performed or should not be performed. In principle, this allows for the dynamic creation of an intelligent checklist that is tailored to a specific situation.

The automated approach of process mining offers an alternative to manually creating healthcare process models, which is hard and tedious work due to the vast amount of domain knowledge that is required. Process mining techniques have been applied to different aspects of healthcare processes in the past (Huang et al. 2014, 2015; Lismont et al. 2016; Mans et al. 2009; Rashid et al. 2017; Rismanchian and Lee 2017; Rojas et al. 2016; Rovani et al. 2015; Wang et al. 2017). However, these applications typically target more standardized processes while using imperative modeling languages as a foundation. Using process mining in combination with declarative modeling languages is actually a relatively new research topic. Maggi et al. (2011) were the first to describe an automated declarative process mining approach, called DeclareMiner. Based on automata, it generates and verifies candidate constraints expressed in the Declare language. By choosing the standard Declare language, the authors limited the approach to discovering only rather simple control flow constraints (i.e., no timed, branched, decision-dependent, data or resource constraints). Later work by Maggi et al. (2012) introduced a two-phase approach to improve the performance by using an Apriori-based first phase to reduce the search space. Maggi, Dumas, et al. (2013) took their approach a step further by adding support for data conditions in a couple of templates of Declare. More recently, a declarative process miner, called TB-MINERful, which can mine branched constraints, was proposed (Di Ciccio et al. 2016). It introduced subsumption and set dominance, two techniques that have also been implemented in DeciClareMiner to reduce the sizes of the mined models. Schönig et al. (2016) also proposed a framework for the discovery of multiperspective

Declare (MP-Declare) models. The work of Schönig et al. focused on an SQL-like on-demand query system. It does not mine a model on its own; rather, an analyst can manually define queries to examine the event log and find useful rules.

DeciClareMiner has the ambitious goal of discovering the declarative control flow of non-standard medical treatment processes, while at the same time also uncovering the medical knowledge governing them. The utilization of Apriori to mine decision-independent rules in the first phase of DeciClareMiner differs slightly from its application in Maggi et al. (2012). We moved away from the basic Apriori approach by differentiating between minimal and non-minimal rules. This customization makes it possible to search for decision-independent rules more efficiently in the second phase of DeciClareMiner. The application of data conditions to a couple of templates of Declare in Maggi, Dumas, et al. (2013) is similar to the activation decisions of DeciClare, although DeciClare supports them for all templates. The miners themselves essentially differ regarding the use of a heuristic search strategy for the second phase, allowing our technique to scale better with input using increasing amounts of data values, as shown in section 3.6.3. The MP-Declare language is even more comparable to DeciClare: both are multiperspective declarative languages based on Declare, but DeciClare distinguishes itself by its additional support from the resource perspective and deactivation decisions. The MP-Declare mining approach requires a significant amount of domain knowledge from the analyst to define useful queries manually, narrowing the application area compared to the automatic mining approach of DeciClareMiner.

3.3. MODELING HEALTHCARE PROCESSES: DECICLARE

The DeciClare language presented in chapter 2 will be used as the process and decision modeling language. Table 11 presents the 9 most used constraint templates of DeciClare, each of which is accompanied by an example that illustrates its meaning and parameters. The parameters I and B define the logical operators that can be used in the parameter expressions of the template as a result

of branching. The letter used refers to the option of using either the inclusive disjunction (I) or both the inclusive disjunction and conjunction (B).

Table 11 omits the timing parameters required for each of the templates, with the exception of the untimed RespondedPresence template, to avoid overloading the table. These are two parameters expressing the time interval in which the constraint applies. For example, `Response(AtLeast(e1, 1), AtLeast(e2, 1), 2, 7)` expresses that if `e1` is executed, then `e2` must be executed at least once after at least 2 time units and at most 7 time units after `e1` was executed. The lower- and upper-bound time parameters are allowed to be 0 and infinity, respectively, making them reduce to the examples presented in Table 11.

DeciClare uses concepts from the DMN decision modeling language to express decision logic. This decision logic is used to define when a constraint can be activated and deactivated. Therefore, a constraint does not have to be satisfied until the activation decision evaluates as true (i.e., decision-dependent). The conditions of a decision can consist of any combination of values of the available data attributes, chained using the logical conjunctive and disjunctive operators (e.g., ‘`a = True`’ and ‘`b = False`’ or ‘`c = “clavicle”`’)). When a constraint has a trivial decision (i.e., one that always evaluates as true), it applies generally (i.e., decision-independent). To illustrate the concept of activation decisions, consider the decision-dependent healthcare rule for the reaction to trauma cases with internal bleeding. The rule would consist of a constraint `Response(AtLeast(Doctor examination, 1), AtLeast(Surgery, 1), 0, 10)` and an activation decision that evaluates as true when it is a trauma case and when the patient exhibits signs of shock and/or heavy local inflammation or pain. This implies that when a patient is a trauma victim and is showing signs of shock and/or heavy local inflammation or pain, surgery must be performed within 10 time units after examination by a doctor. The constraint will be activated during such an examination, prompting emergency surgery.

Class	Template	Example
<i>Existence</i>	AtLeast(I, n)	AtLeast(Apply bandage or Apply cast, 2): the sum of the occurrences of Apply bandage and the occurrences of Apply cast must be at least 2
	AtMost(B, n)	AtMost(Examine patient and Take X-ray, 3): the number of co-occurrences of Examine patient and Take X-ray must be at most 3
	First(I)	First(Register patient, Examine patient): either Register patient or Examine patient must be executed before any other event can be executed
	Last(I)	Last(Apply sling or Examine patient): either Apply sling or Examine patient must be executed as the final event
<i>Relation</i>	RespondedPresence(B ₁ , B ₂)	RespondedPresence(AtLeast((Take X-ray and Apply sling) or Apply bandage, 1), AtLeast(Examine patient or Perform surgery, 1)): if Take X-ray and Apply sling or just Apply bandage are executed at least once, then Examine patient or Perform surgery must also be executed at least once somewhere in the trace (before, during or after)
	Response(B ₁ , B ₂)	Response(AtLeast((Take X-ray and Apply sling) or Apply bandage, 1), AtLeast(Examine patient or Perform surgery, 1)): if Take X-ray and Apply sling or just Apply bandage are executed at least once, then Examine patient or Perform surgery must also be executed at least once at some time after Take X-ray and Apply sling or after Apply bandage
	ChainResponse(B ₁ , B ₂)	ChainResponse(AtLeast((Take X-ray and Apply sling) or Apply bandage, 1), AtLeast(Examine patient or Perform surgery, 1)): if Take X-ray and Apply sling or just Apply bandage are/is executed at least once, then Examine patient or Perform surgery must also be executed at least once immediately after Take X-ray and Apply sling or after Apply bandage
	Precedence(B ₁ , B ₂)	Precedence(AtLeast((Take X-ray and Apply sling) or Apply bandage, 1), AtLeast(Examine patient or Perform surgery, 1)): Take X-ray and Apply sling or just Apply bandage can be executed only after Examine patient or Perform surgery has already been executed at least once
	ChainPrecedence(B ₁ , B ₂)	ChainPrecedence(AtLeast((Take X-ray and Apply sling) or Apply bandage, 1), AtLeast(Examine patient or Perform surgery, 1)): Take X-ray and Apply sling or just Apply bandage can be executed only immediately after Examine patient or Perform surgery has been executed at least once

Table 11. DeciClare constraint templates used in chapter 3

3.4. RESEARCH METHODOLOGY

This paper applies the Design Science Research Methodology (Hevner et al. 2004; Peffers et al. 2007) to develop a process mining tool (i.e., the design artifact) that can mine a declarative process and decision model from data logged in the information systems of an organization. The requirements for the design artifact are as follows:

- It should be able to handle an event log as input. We will assume that the event log is available as an XES-file. The standard XES should be fully supported, as well as (custom) extensions to add extra data and resource definitions.
- It should offer parameters to handle noise in event log and set the granularity of the output models.
- The output should be a mixed-perspective process and decision model.
- It should be able to run in practical timespans. We defined this as support for ‘given an amount of time, give me the best possible model that you can mine’.

The scope is set to the 9 constraint templates from the functional and control-flow perspectives (including negation and branched templates) (see Table 11) and activation decisions from the data perspective. These are certainly the most useful and therefore could be regarded as the minimum viable product. Also, most of the data and resource templates of DeciClare cannot be mined automatically using a traditional event log because these logs do not have the ability to store the required data (e.g., the number of instances available of a certain resource at a certain time, authorization templates etc.).

The evaluation will focus on the last two requirements, as the first two are mostly default requirements for any process mining tool. The resulting tool will be compared with a naïve brute-force strategy. The comparison will feature the completeness of the model and the time required to discover it.

3.5. DISCOVERING HEALTHCARE PROCESS MODELS: DECICLAREMINER

The process mining algorithm that can be used to discover DeciClare models is called DeciClareMiner. Fig. 12 presents a high-level flow diagram of the proposed DeciClareMiner algorithm. The input, i.e., an event log, is first entered into the phase 1 algorithm (described in section 3.5.1). In the first phase, all rules that always apply according to the log, independent of

Fig. 12. High-level flow diagram of the proposed DeciClareMiner algorithm

decisions, are mined (e.g., $\text{AtLeast}([\text{Clinical examination of patient}], 1)$). These rules have a trivial activation decision (i.e., the decision always evaluates as true). The outcome of this phase is twofold: a set of decision-independent rules and a set of rules that did not make the cut but satisfy certain criteria (e.g., $\text{Response}(\text{AtLeast}(\text{Perform surgery}, 1), \text{AtLeast}(\text{Apply cast}, 1))$ with a trivial activation decision, is correct for 493 traces and violated in 4018 traces). The former will be stored temporarily pending the completion of the rest of the algorithm. The latter, together with the original event log, will serve as the input for the phase 2 algorithm (described in section 3.5.2). The second phase mines rules that only apply in specific situations and thus are dependent on certain (activation) decisions (e.g., $\text{Response}(\text{AtLeast}(\text{Perform surgery}, 1), \text{AtLeast}(\text{Apply cast}, 1))$ activateIf[[Fractured bone = Wrist, More than 16 yrs old = true], [Fractured bone = Wrist, Extensive damage to arteries or nerves = true], [Open or complex fracture = true, Fractured bone = Wrist], [Fractured bone = Forearm], [Fractured bone = Wrist, Periosteum torn = false]]). The outcome of this phase will complement the already mined decision-independent rules with decision-dependent rules to form the eventual outcome of DeciClareMiner: a DeciClare model of the process contained by the given event log (with a given minimal confidence and support level). Table 12 presents the definitions of important concepts that are used throughout this paper.

3.5.1. Phase 1: Mining decision-independent rules

The algorithm used to mine rules that always apply is largely inspired by the Apriori algorithm (Agrawal and Srikant 1994). The Apriori algorithm was originally designed to mine frequent itemsets in the context of association rule mining. An itemset is called frequent if it occurs at least

<i>Concept</i>	<i>Definition</i>
<i>Constraint template</i>	A type of constraint (e.g., AtLeast). A template has a specific interpretation and can have certain parameters.
<i>Constraint</i>	A (mandatory or optional) limitation of the universe of discourse represented by the process model (e.g., AtLeast(A, 1, 0, 5)). A constraint is an instantiation of a template, with values given for all necessary parameters.
<i>Constraint activation</i>	A constraint for which the activation decision evaluates as true and, when it is a relation constraint, the condition side of the constraint is satisfied. This says nothing about the satisfaction of the actual constraint.
<i>Conforming activation</i>	A constraint that is activated <i>and</i> satisfied by a certain (partial) trace (e.g., consequence side of an activated relation constraint is satisfied).
<i>Violating activation</i>	A constraint that is activated <i>and</i> violated by a certain (partial) trace (e.g., consequence side of an activated relation constraint is not satisfied).
<i>Vacuous satisfaction</i>	A constraint that is not activated and thus holds trivially (i.e., it is not violated).
<i>Rule</i>	A constraint in the context of a certain event log. A rule contains a constraint and the corresponding statistics related to the application of the constraint in the given event log (e.g., confidence and support of the constraint).
<i>Confidence</i>	The number of conforming activations of a rule (i.e., vacuous satisfactions are ignored) divided by the total number of activations of the rule. For example, a rule that has 493 conforming activations and 4018 violating activations has a confidence of 10.9% ($= 493/(493+4018)$)
<i>Support</i>	The total number of traces in which a rule is activated (i.e., vacuous satisfactions are ignored) divided by the total number of traces in the event log. For example, a rule that has 493 conforming activations and 4018 violating activations in a log of 5000 traces has a support of 90.2% ($= (493+4018)/5000$)

Table 12. Definitions of important concepts

a certain number of times (i.e., support is higher than the given threshold). Apriori uses a breadth-first complete-search approach that starts from the smallest frequent itemsets and works its way up to the largest available frequent itemsets. The key principle used by the Apriori algorithm is that any subset of a frequent itemset must also be a frequent itemset. Thus, for example, a set containing items A and B cannot be frequent if either (or both) item A or item B on its own is not frequent. Instead of evaluating every possible combination of items, only those that have a chance of being a frequent itemset are evaluated.

Although declarative rules can be perceived as more complex association rules, identifying declarative rules in an event log is quite different from identifying frequent itemsets in a transactional database. First, the evaluation of a rule is much more complex compared to simply counting occurrences in transactions. Each constraint template has a different logic that needs to be

evaluated. Additionally, we are interested in not only the support of a rule in the event log but more so its corresponding confidence. Second, DeciClare does not have only templates connected to one logical expression but also templates that are connected to two logical expressions (i.e., relation templates). Finally, several templates also have an integer parameter (i.e., a bound) in addition to one or two logical expressions. Therefore, to use the Apriori algorithm, a mapping of its core concepts needs to be defined (see Table 13). The pseudocode that results from the mapping of the Apriori pseudocode is presented in Listing 1. In the remainder of this section, the non-trivial mapping steps will be explained in more detail.

	<i>Apriori for frequent itemsets</i>	<i>Apriori for declarative rules</i>
<i>Input</i>	A transactional database and a minimal support threshold	An event log, a minimal support threshold and a minimal confidence threshold
<i>Element</i>	Itemsets	Rules connected to logical expressions consisting of activities, events, resources or data elements
<i>Goal</i>	Identify all itemsets with a support level at least equal to a given value (= frequent itemset)	Identify all minimal declarative rules with confidence and support levels at least equal to given values (= frequent minimal rules)
<i>Leveraged property</i>	Any subset of a frequent itemset is also frequent	Any rule connected to a logical expression that is a subset of the logical expression connected to a minimal rule (same template) is also minimal
<i>Iteration principle: initialization</i>	Start from all possible itemsets with size 1	For each type of constraint template, start from constraints with logical expressions of size 1 (i.e., no logical operators)
<i>Iteration principle: joining</i>	Increase the sizes of the itemsets by joining all sets that differ by one element	Increase the size of the connected logical expressions by joining all rules based on the same constraint template that do not have a 100% confidence level and have logical expressions that differ by just one element
<i>Iteration principle: optimization</i>	Only evaluate itemsets that can have a support level at least equal to a given value	Only evaluate rules that can reduce the number of violations while still having a support level at least equal to a given value

Table 13. Mapping of the Apriori algorithm to our problem domain

3.5.1.1. Leveraged property

In the mapping of the goal of the algorithm, we introduced the notion of a minimal rule. A rule is minimal if no subset of the connected logical expression results in an equal or lower number of violations. In terms of the goal of the algorithm, discovering a non-minimal rule gets us no closer to a rule with a 100% confidence level than at least one of the previously discovered rules, with a

Input: set F_1 containing all minimal rules with logical expressions of size 1, a support threshold, a confidence threshold and a maximal branching level m
Output: set F containing the union of sets $F_1, F_2, \dots$
 For $k = 1$ to m or until F_k is empty
 1. Generate candidates set C_{k+1} containing rules with logical expressions of size $k+1$ by joining set V_k , which contains all rules in set F_k with at least one violation, with itself
 2. Prune C_{k+1} by removing all rules that have a logical expression that contains the logical expression of a non-minimal rule of size k based on the same constraint template
 3. Evaluate the rules in C_{k+1} , and put the rules with at least one conforming occurrence into set F_{k+1}

Listing 1. Pseudocode of the first phase of the algorithm

logical expression that is a subset of the non-minimal rule, already did. The non-minimal rule thus offers no additional value over a subset rule that has at most an equal number of violations; at the same time, it is more complex. Consider, for example, an event log with three traces: A, AB and ACD. Evaluating response constraints with only event A in the condition side logical expression results in Fig. 13. The first step calculates the confidence level of rules with a consequence-side logical expression size of 1 (e.g., '+1' = one conforming occurrence, and '-2' = two violating occurrences, resulting in a 33% confidence level). In the following step, these are joined pairwise to create rules with a consequence-side logical expression size of 2. Response(AtLeast(A, 1), AtLeast(B or C, 1)) is a minimal rule because it has one less violation than any of its subsets, in this case, than its parent rules Response(AtLeast(A, 1), AtLeast(B, 1)) and Response(AtLeast(A, 1), AtLeast(C, 1)). The same can be said of Response(AtLeast(A, 1), AtLeast(B or D, 1)). However, Response(AtLeast(A, 1), AtLeast(C or D, 1)) is not a minimal rule because it has the same number of violations, namely, 2, as both of its parents Response(AtLeast(A, 1), AtLeast(C, 1)) and Response(AtLeast(A, 1), AtLeast(D, 1)). Based on the property of Table 13, we can already conclude that the rule Response(AtLeast(A, 1), AtLeast(B or C or D, 1)) is non-minimal, as it contains the non-minimal subset rule Response(AtLeast(A, 1), AtLeast(C or D, 1)).

Fig. 13. Example of the evaluation of the response template

3.5.1.2. The iteration principle: optimizations

The three steps of the algorithm in Listing 1 are repeated until the constraints are branched up to the maximal branching level or until no new constraints are found in the previous execution of these steps. The leveraged property allows for three optimizations to be applied in this iteration: the join criteria, the join step and the prune step. The latter does not differ from the corresponding step in Apriori, but the former two require slightly more attention.

3.5.1.3. The join criteria

Similar to Apriori, a join can be applied to two rules if the logical expressions differ by exactly one element. However, an additional criteria states that only rules with a confidence level lower than 100% will be joined. This is because joining two rules, of which at least one has a confidence level of 100%, will not result in anything better than the two original rules. For example, joining rules $\text{Response}(\text{AtLeast}(A, 1), \text{AtLeast}(B, 1))$ and $\text{Response}(\text{AtLeast}(A, 1), \text{AtLeast}(C, 1))$ will result in a rule $\text{Response}(\text{AtLeast}(A, 1), \text{AtLeast}(B \text{ or } C, 1))$, which will be explained below. If it is already known that event A is always followed by event B (confidence level of 100%), then additionally stating that A is always followed by event B or C (i.e., the child rule) offers no new information.

3.5.1.4. The join step

Rules with a logical expression of size k are joined to create rules with a logical expression of size $k+1$ in the iteration steps of the algorithm. However, the elements of the logical expressions can be connected with either the inclusive disjunction or the logical conjunction operator. Therefore, the influence of joining these logical expressions using either operator on the number of violations and the support level is investigated for each template when the leveraged property from Table 13 is applied (see Appendix E for the details). As a result, certain rules (e.g., OR join for the condition side of relation rules) will never be joined, as they will never lead to a new minimal rule being found.

However, the AtLeast and AtMost templates have an additional numerical limit associated with them. This makes it very inefficient to apply the Apriori candidate generation step to these templates, as this would require the generation of all possible event sets of size k with all possible limits (= a candidate set of infinite size). Instead, our algorithm makes an exception regarding the Apriori-inspired iteration for these templates and will generate all possible logical expressions and, for each logical expression, calculate the associated limit based on the event log directly. To increase the efficiency, these templates are evaluated together, which makes it possible to generate all rules with a meaningful limit of both templates for some logical expression in one pass of the event log with no performance hit compared to evaluating one rule of these templates (similar optimization is also used for the other templates). For example, suppose A occurs twice and three times, respectively, in an event log with two traces. With a support level of more than 50%, two meaningful rules can be mined in one step: $\text{AtLeast}(A, 2)$ and $\text{AtMost}(A, 3)$.

3.5.1.5. Running example

To illustrate how the first phase works, let us consider the following example for the consequence side of the positive Response template (Fig. 14). We have an event log of a process with the following traces: AB, AAB, CAC and ABABC. The confidence threshold will be set at 80%, and the support threshold at 50%. The algorithm starts by creating every Response constraint possible with a condition and consequence expression of size 1. The candidates are subsequently evaluated by calculating their confidence and support levels. The first candidate rule has a confidence and

Fig. 14. Running example of the first phase (minimal rules in black; non-minimal rules in grey)

support level above the given thresholds and will therefore be added to the result set. Note that the first five rules are minimal rules, as they have no subset rules, but only the first rule is a frequent minimal rule. Next, a new set of candidate rules is generated by joining all compatible candidate rules from the previous candidate set. Joining the consequence side of the positive Response template will be done using only the OR operator (see Appendix E). Additionally, the last two rules from the candidate set from iteration 1 are not joined since the latter rule has a confidence of 0% (i.e., non-minimal). The result is a new candidate set with two new candidate rules. Since all rules from the candidate set of the previous iteration are minimal, no candidates can be pruned at this point. The subsequent evaluation of the candidates yields one frequent minimal rule and one non-minimal rule (i.e., the second candidate has as many violations as the fourth candidate rule, which is one of the parent rules, from iteration 1). The first candidate will be added to the result set to give us two frequent minimal rules. The algorithm stops at this point, as no candidates of size 3 can be generated.

3.5.1.6. The output of the first phase

This first phase results in a set of decision-independent rules with a confidence and support level equal or higher than the given thresholds and logical expressions of a size equal to or less than the given maximal branching level (top output in Fig. 12). As a waste product, it also generates all minimal rules no matter the confidence and support levels. This set of ‘waste’ rules will be used to optimize the second phase of the algorithm (bottom output in Fig. 12). With the exception of the AtLeast, AtLeastChoice, AtMost and AtMostChoice templates, all decision-dependent rules can be represented by one of these waste rules combined with some decision. Therefore, the constraints of all these minimal waste rules with a confidence level of less than 100% and a support level higher than the support threshold (as the support level of a rule can only drop by adding an activation decision) that were generated during this first phase will be used as seed constraints for the second phase. In the running example, this would be the first five candidate rules from the first iteration. As a result, the second phase will not have to generate any constraints and can focus on only the

decision logic. This will significantly reduce the complexity of defining a search strategy for finding decision-dependent rules and substantially reduce the size of the search space.

For the AtLeast and AtMost templates, only the AtMost rules with 100% confidence levels are used as input, as these express the boundaries within which it is meaningful to search for the decision-dependent rules of these templates. For example, from the rule $\text{AtMost}(A, 3)$, with a confidence level of 100%, we can deduce that any meaningful decision-dependent AtMost rule with the same logical expression must have a bound of 0, 1 or 2 and that any meaningful decision-dependent AtLeast rule with the same logical expression must have a bound of 1, 2 or 3.

3.5.2. Phase 2: Mining decision-dependent rules

In the second phase, decision-dependent rules are generated. Our assumption is that for loosely framed KiPs, the number of possible activities will be much lower than the number of possible combinations of data elements on which the activation decisions are based. Therefore, while the first phase uses a complete search algorithm, the second phase uses a heuristic approach to search for meaningful rules from the vast set of possible combinations of data values and seed constraints from the first phase.

For problems with a search space that is too large to go over completely, heuristics can offer ‘good enough’ solutions by sampling the search space. A subset of these heuristics, called genetic algorithms, mimic evolutionary processes that utilize natural selection. These processes are non-deterministic but have been proven and tested by nature to be very flexible and efficient. The heuristic that we use belongs to the class of genetic algorithms that use the crossover, mutation and selection operators to solve constrained or unconstrained optimization problems (Mitchell 1996). The approach requires the problem to be redefined in terms of finding the best genetic codes (i.e., individuals) according to some fitness function. An individual consists of a subset of the available genes, which are the features that can make the individual a better or worse fit as a solution for a problem compared to other individuals. A population of these individuals is recombined and mutated over multiple iterations to allow the fitness of the population to evolve by sharing the genes

of the better individuals. Table 14 presents a mapping of the general genetic approach to our problem domain.

	<i>Genetic decision-dependent rule miner</i>
<i>Input</i>	Seed rules, the available data elements, an event log, an intermediate population size, a minimal support threshold and a minimal confidence threshold, the number of new seeds to be added in each iteration, a fitness function and a recombination and mutation frequency
<i>Element</i>	A decision-dependent rule, with its activation decision serving as its genetic code
<i>Fitness</i>	A measure of how well a decision-dependent rule reflects the underlying process of the event log
<i>Goal</i>	Find the best decision-dependent rules with a confidence and support above the thresholds
<i>Operations</i>	To find better decision-dependent rules, changes are introduced by recombining and mutating candidates; this allows good decision logic to be shared across the population
<i>Iteration principle</i>	The best decision-dependent rules of the previous iteration and some new seed rules with random decisions form the population of the next iteration; this offers a balance between searching in a specific direction (i.e., of previously found good decision-dependent rules) versus in all directions (i.e., new good decision-dependent rules)

Table 14. Mapping of the genetic approach to our problem domain

The search problem of finding the best decision-dependent rules can be redefined as finding activation decisions for the seed constraints from the first phase. An individual is a combination of a seed constraint and a non-trivial activation decision. The decision values, which can form an activation decision, can be regarded as the genes of the genetic code of individuals. The basic approach is illustrated in Fig. 15. The algorithm starts by combining seed constraints, produced in the first phase, with random activation decisions (e.g., ‘Has fever?’ = false and ‘headache’ = true) to create the initial population of individuals. The individuals in the initial population are then evaluated and sorted according to the fitness function. The recombination operator is subsequently

Fig. 15. Overview of the second phase of the algorithm

used to create a population of children of this population. The higher the fitness of a rule, the higher the chances of being selected as one of the two parent rules of a new child rule. The child gets half of the genes of each parent, with some random mutations being applied to the genes based on the mutation rate. In the next step, this child population is evaluated and sorted according to the fitness function. The best children are used to create a new edit population, together with a given percentage of new seeds in each iteration. This edit population is used to start the next iteration until no new individuals are added to the result population during a given number of iterations or the given maximal amount of search time has been exceeded. Individuals that satisfy the confidence threshold are integrated into the result population during the two evaluation and sort steps of each iteration. Note that the support threshold is ignored at this point and will only be introduced during the post-processing steps, because up to that point each disjunct activation decision for a certain constraint is discovered separately and the support of the constraint can only be calculated by combining all these disjunct pieces into one rule during post-processing.

3.5.2.1. Fitness Function

To measure the fitness of an individual, we use the weighted average of four partial scores (all expressed as percentages):

- I. A score that expresses the increase of the trace conformance (i.e., number of conforming traces divided by the total number of traces for which the rule was activated) of an individual compared to the decision-independent seed rule on which the individual was based.
- II. A score that expresses the generality of the activation decision of an individual. The score is higher for individuals with an activation decision that uses less decision elements.
- III. A score that punishes individuals that stay dominant in the population for too many iterations. Each individual has a generation, i.e., the number of recombination operations it has undergone. An individual created from a seed constraint and a random decision has 0 as generation and an individual created in a recombination step has one plus the highest

generation of its parents as generation. The punishment is negligible for younger generations but rises exponentially. This ensures that the search direction stays sufficiently broad.

- IV. A score that expresses the potential of the individual to increase the diversity of the result population. A frequency map, which stores the number of times a certain constraint (with its specific parameter values, ignoring the activation decision) has already been found with satisfactory confidence and support levels, is kept. An individual with a constraint that is part of an already found individual will progressively receive a lower and lower score as the frequency of found individuals with the constraint rises. This ensures that other search directions will be explored more frequently as frequent search directions are being exhausted.

3.5.2.2. Optimization

The obvious choice would be to run the second phase algorithm for all templates at once. However, because we grouped certain templates together to increase the efficiency of the evaluation step, the fitness score of such a group of constraints cannot be fairly compared to that of other groups based on other templates. For example, a relation group evaluates the RespondedPresence, Response, ChainResponse, Precedence, and ChainPrecedence templates as well as their negative counterparts for a certain set of required parameters, while the bounded group does the same for the AtLeast and AtMost templates. Its score for the partial fitness measure I is a weighted combination, based on the subsumption hierarchy (Di Ciccio and Mecella 2015), of the scores of all rules found for this group. However, for the bounded group, there will always be at least one rule found with a confidence of 100% because of how they are evaluated: $\text{AtMost}(\text{AE}, x)$, with x being the maximum number of times the activity expression ‘AE’ occurred in any trace of the given log (see end of section 3.5.1). Therefore, the bounded group fitness will generally be higher than the relation group fitness. Hence, the second phase algorithm will be executed separately for each group of templates. The size of the intermediate populations for each group is set relative to the number of available seed rules for that group. Each iteration of the global algorithm for the second phase executes one iteration for each

group, unless the group has already converged. Finally, the result populations of each group are merged.

3.5.2.3. Running example

Fig. 16. Running example of the second phase

Fig. 16 presents a running example of how the second phase would be performed for the *AtLeast* template (i.e., ignoring the *AtMost* template of the group). The confidence (c) and support (s) thresholds are set to 100% and 50%, respectively. For simplicity, we set the size of the intermediate populations to just two individuals, with four children, and consider only one seed constraint, namely, *AtLeast*(A, 1) (i.e., ‘A is executed at least once’), with its corresponding decision-independent rule (i.e., *AtLeast*(A, 1) with a trivial activation decision) having a confidence of 60% and a support of 100%. One can assume that the bound parameter does not change. Three data attributes are used in this example: x (Boolean), y (Boolean) and z (categorical with values: ‘1’, ‘2’ and ‘3’). In each iteration, one new seed is added to the new edit population.

In the first step of the second phase, an initial population is created by combining the seed constraint with random decisions based on the available data attributes. The constraints in the initial population are then evaluated and added to the edit population, which in turn is sorted according to

the fitness function. In the next step, the child population is created by recombining the rules in the sorted edit population with random splits of their genes. This process also introduces random mutations in the genes of the children (grey). The child population is then sorted, with one rule making it into the result population. The best rule of the sorted child population is also transferred to the new edit population for the second iteration, along with a new seed constraint combined with another random decision. The process continues until it converges, or the time limit is reached.

3.5.2.4. The output after the second phase

After the iteration steps of the genetic algorithm, three post-processing steps are performed to improve the combined results from both phases. The first step collects all decision-independent rules that satisfy the confidence and support thresholds from the results of the first phase while also removing subsumed rules according to the subsumption hierarchy (e.g., keep `ChainResponse(A, B)` and remove `Response(A, B)`, as the former is a stricter version of the latter). The second step merges the decision logic for rules from the results of the second phase with the same constraint but different activation decisions. This step also removes all subsumed rules from these results. The last step combines both results and removes redundant rules (e.g., `First(A)` makes `AtLeast(A, 1)` redundant).

3.6. EVALUATION

The DeciClareMiner process discovery algorithm is implemented (fully single-threaded) to conduct an evaluation. The implementation of DeciClareMiner used in this paper can be found at <https://github.com/stevmert/DeciClareMinerV11/>. The event log and case description (.../models/) as well as a summary of the results of the experiments (.../results/) can also be found at the above link. For the actual results of each of the experiments (ca. 1.13 GB of compressed zip-data, mostly in .CSV format), please contact the author.

3.6.1. Dataset

In the previous chapter (also see Mertens et al. 2017), the healthcare process for treating arm-related fractures was modeled using the direct instantiation of the DeciClare abstract syntax metamodel. The process entails the registration, diagnosis and treatment phases for patients with one or more (possible) fractures of a finger, hand, wrist, forearm, upper arm, shoulder, and/or collarbone. This DeciClare process model was created based on the available literature and by interviewing doctors and nurses that participate in this healthcare process in the emergency and orthopedic department of several hospitals in Ghent, Belgium. This fictional (i.e., this is not an isolated process in real-life) but realistic healthcare case will be used for the evaluation of the DeciClareMiner process discovery algorithm.

We first generated an event log based on the DeciClare model of the arm fracture case. We identified 2072 unique realistic combinations of control flow and data that fit the model correctly (i.e., no noise). This is of course a simplification, as the model theoretically allows an infinite number of unique combinations. Additionally, we chose to exclude some of the activities from the original model (i.e., all prescriptions of medication and physiotherapy) for simplicity. The resulting event log uses 9 data attributes, 1 categorical (7 categories) and 8 Boolean, and 9 activities and was created by adding each unique combination of control flow and data at least once, but possibly multiple times (at random), to the log until it contained exactly 5000 traces.

As an example of the kind of rules that correspond to the arm fracture case, consider the following two relatively complex examples. First, one rule yields all situations in which surgery will need to be performed at least once sooner or later: `AtLeast(Perform surgery, 1, 0, -1) activateIf[[Open or complex fracture = true], [Fractured bone = Upper arm, No improvement in 3 months = true], [Extensive damage to arteries or nerves = true], [Fractured bone = Upper arm, Both arms broken = true], [Periosteum torn = true, Fractured bone = Wrist]]`. The first part represents the constraint (i.e., surgery must be performed at least once), while the second part describes its activation decision (i.e., all the possible data combinations for which the constraint applies). This rule is activated in

4511 of the 5000 traces in the arm fracture event log and thus is regarded as frequent behavior. The second example represents less frequent behavior contained by the event log, i.e., in 493 of the 5000 traces. It describes the situations in which a cast needs to be applied after surgery is performed: `Response(AtLeast(Perform surgery, 1, 0, -1), AtLeast(Apply cast, 1, 0, -1), 0, -1) activateIf[[More than 16 yrs old = true, Fractured bone = Wrist], [Extensive damage to arteries or nerves = true, Fractured bone = Wrist], [Open or complex fracture = true, Fractured bone = Wrist], [Fractured bone = Forearm], [Periosteum torn = false, Fractured bone = Wrist]]`. The decision logic associated with both of these example rules is presented in its most general and precise form. Of course, subsets of this decision logic exist that describe only parts of the general behavior (i.e., missing disjunctive decision logic or overfitting in a few specific cases) and can therefore uniquely identify only a subset of the traces of these rules.

3.6.2. Benchmark creation

To analyze the DeciClare models mined using DeciClareMiner, benchmarks were created for several branching levels. A complete search algorithm was used to mine reference models that could serve as benchmarks for branching levels 1 and 3 and for unlimited branching. Table 15 highlights some important numbers from the application of the complete search algorithm, which were executed one by one on a single core of an Intel E5-2680v3 with the confidence threshold set to 100% (as there is no noise in the log) and no support threshold (as most of the 2072 variations occur only once or twice due to limiting the log to 5000 traces). From here on, we will refer to the models mined with the complete search algorithm as the ‘complete models’. These models do not contain any subsumed or redundant rules, as these are filtered out automatically. Note that the complete model with unlimited branching is not the same as the model from previous work (Mertens et al. 2017), because the complete model is based on an event log containing just a subset of the behavior of the original model. Each seed rule needed to be checked for all 279935 possible data combinations for both branching levels. This vast search space is the result of the 9 data attributes used in the arm fracture event log and is still very limited compared to real-life cases. Compared to branching level

1, the search space increases by an additional 736% for a branching level of 3 and by 1475% for unlimited branching. This supports the assumption made at the start of section 3.5.2, based on which a heuristic for the second phase of the algorithm was adopted instead of a complete search approach. The increased search space for branching levels higher than 1 make it infeasible to calculate a complete search model in real-life cases.

To calculate how the models mined using DeciClareMiner stack up to the benchmarks, two criteria are used: the percentage of traces uniquely identified, and the percentage of constraints identified by the model as a whole compared to the corresponding reference model. For example, an overfitted version of the first example rule from section 3.6.1 might only be able to uniquely identify 3000 traces. This means that for this rule, the mined model would score 3000 out of a possible 4511 for the first criteria and 1 out of 1 for the second criteria. However, if the mined model does not contain any version of this rule, it would score 0 out of a possible 4511 for the first criteria and 0 out of 1 for the second criteria for this rule. The scores are calculated for each rule in the corresponding reference model, are compared to the rules in the mined model, and together serve as the score for the mined model.

Branching level	Time	# of seed rules for Phase 2	# of rules mined in Phase 2	Total # of rules mined in both Phase 1 and 2
1	49 h 39 min	56	129	268
3	353 h 24 min	412	973	1177
Unlimited	734 h 42 min	826	3989	4205

Table 15. Application of the complete search algorithm for different branching levels

3.6.3. Results

Using the arm fracture case event log as input, DeciClareMiner was executed using the same single core of an Intel E5-2680v3 with the same confidence and support thresholds as for the benchmark creation (100% and 0%, respectively), with different settings for the branching level and amount of search time for the second phase. For each branching level and search time combination, DeciClareMiner was executed many times (i.e., # of experiments column in Table 16). This was

necessary because the second phase of the algorithm is not deterministic, which means that the results of the miner can vary. Based on many runs of the miner on the same settings, we were able to calculate the average coverage of the mined models using the predefined branching level and search time.

Branching level	Search time P2	% of traces identified	% of constraints identified	# of experiments	Speed-up factor
1	10 min	93.32% ($\sigma = 1.42\%$)	98.72% ($\sigma = 0.84\%$)	271	278
	20 min	97.64% ($\sigma = 0.52\%$)	99.56% ($\sigma = 0.48\%$)	409	145
	40 min	99.05% ($\sigma = 0.19\%$)	99.82% ($\sigma = 0.34\%$)	117	74
3	10 min	69.22% ($\sigma = 1.13\%$)	81.38% ($\sigma = 1.27\%$)	270	1466
	20 min	78.32% ($\sigma = 1.71\%$)	89.39% ($\sigma = 1.28\%$)	410	830
	40 min	87.29% ($\sigma = 1.15\%$)	94.33% ($\sigma = 0.70\%$)	117	462
Unlimited	10 min	50.57% ($\sigma = 1.22\%$)	58.89% ($\sigma = 1.23\%$)	364	2229
	20 min	59.55% ($\sigma = 1.59\%$)	67.85% ($\sigma = 1.78\%$)	572	1312
	40 min	68.2% ($\sigma = 1.04\%$)	77.41% ($\sigma = 1.16\%$)	307	752

Table 16. Results from the mining experiments for different settings

The power of the chosen heuristic is illustrated by the fact that DeciClareMiner found 99% of all the constraints and 93% of the total decision logic in just 10 minutes (see Table 16). The benchmark needed 49 hours and 38 minutes to calculate the complete search model. If we assume that the complete search algorithm mined the results evenly over this runtime (i.e., 0.03% per minute), it would have mined 93.32% of the total decision logic after 46 hours and 20 minutes. Then, DeciClareMiner speeds this up by a factor of 277 (i.e., 9.33% per minute). Fig. 17 illustrates how the intermediate populations evolve during the execution of the algorithm (please note that the results in Fig. 17 are a bit worse than the actual performance of the algorithm due to the added overhead of having to store the intermediate population after each iteration). The second phase of DeciClareMiner does most of the work early on due to the innate characteristics of the genetic approach. After just 13 iterations (~1 minute), 88% of all the constraints and 61% of the total decision logic had already been found. The scores further approach the model solution with each additional iteration, though the speed drops significantly at the same time. Almost 60% of the model had already been found during the first minute of execution (speed-up factor of 1740). However, although the results further improve with a longer runtime, to 98% and 99% for 20 and 40 minutes,

respectively, the speed-up factor drops significantly at the same time to 145 (4.88% per minute) and 74 (2.48% per minute), respectively.

For branching levels 3 and unlimited branching, the mined models still obtain respectable averages of 87% and 68% of the benchmark model in 40 minutes. Although the absolute speed declines for higher branching levels (2.18% and 1.70% per minute, respectively, during the 40-minute search), the relative speed-up factor remains high compared to the runtimes for a complete search approach. DeciClareMiner speeds up the discovery of 87% of the model by a factor of 462 for branching level 3 and of 68% of the model by a factor of 752 for unlimited branching.

Fig. 17. Evolution of the intermediate population scores

3.6.4. Discussion

As illustrated in the evaluation, the miner presented in this paper is a promising step towards modeling and capturing the knowledge of real-life healthcare processes and, more generally, of loosely framed KiPs. DeciClareMiner can efficiently mine declarative process models containing decision-independent and decision-dependent constraints. It was able to produce a model that represents 93% of episodes of care with atomic constraints in a relatively short time (10 minutes). This is considered a major improvement compared to the 50 hours it took to calculate the 100%-episode model using an exhaustive search approach. The main building blocks of the DeciClareMiner algorithm, i.e., the Apriori-inspired first phase algorithm and the genetic algorithm

of the second phase, can address rising complexity very well. This behavior was expected due to the selection of a genetic algorithm as the basis for the second phase. Similar to the natural processes on which genetic algorithms are based, genetic algorithms offer good performance when faced with high complexity. In addition, although not a part of this evaluation, they should also be able to handle varying levels of noise in the event log, as these techniques are considered relatively noise-resistant.

The downside of using a heuristic to search for decision-dependent rules is that we cannot guarantee that every detail of the complete search model will ever be found. As Fig. 17 shows, the algorithm will quickly find most of the complete search model but eventually slows down more and more as it approaches 100%. We believe that this is a necessary trade-off to apply this kind of technique to the domain of healthcare in real-life environments. The calculation time of the unbranched complete model of the arm fracture log already required more than 30 days of nonstop mining. In real-life settings, the cases are much more complex. The event log used in the evaluation has a size of just 26 MB, whereas event logs for real-life cases can easily be hundreds of MB (e.g., the MIMIC-III clinical database (Johnson et al. 2016) contains 43 GB of data). Clearly, the available processing power and time are going to be the limiting factors. Additionally, the domain is very dynamic, as medical knowledge evolves continuously. In this setting, it is better to have a very good approximation of the complete search model in reasonable time than a complete search model that is already at least partially obsolete by the time it is available. We believe that the efficiency offered by DeciClareMiner is necessary for the adoption of process mining in real-life healthcare organizations and, more specifically, in non-standard medical treatment processes.

DeciClareMiner can also be applied when the event log is not complete, as in our evaluation and in almost every practical case. However, this requires a subsequent evaluation of the resulting set of rules by domain experts to relax possibly overfitted rules. Because DeciClareMiner calculates the whole subsumption hierarchy, this relaxation is just a matter of selecting an already available rule that is lower in the hierarchy (e.g., if the event log only contains episodes of care that execute activity

B immediately after A, which is not that strict in real life, the mined rule $\text{ChainResponse}(A, B)$ can be relaxed to $\text{Response}(A, B)$). The use of a declarative language makes the evaluation of the rules by domain experts relatively easy, as each rule can be evaluated in isolation.

The availability of DeciClare models of loosely framed KiPs can enable many new applications at the management level. For example, consider the checklists mentioned in section 3.2. They are already being used throughout the healthcare sector at an operational level to reduce the number of human errors that occur due to the innate and overwhelming complexity of non-standard medical treatment processes. Declarative process models can be used to generate these checklists on-demand during the execution of a process. The dynamic checklists can give an overview of the activities that still need to be performed, those that should not be performed and those that are optional. The decision logic in a mined DeciClare model can help produce tailor-made checklists for patients currently being treated, taking into account the specific patient characteristics and progression of treatment. Of course, it would be advisable to have the mined model validated by experts first. The treating physicians could also perform this validation on-the-fly, as the knowledge mined by DeciClareMiner represents only a small portion of their extensive domain knowledge.

3.7. LIMITATIONS

The current implementation is already rather efficient, as the first phase and each iteration of the second phase of the algorithm each have a linear asymptotic time complexity (i.e., $O(n)$) in relation to the number of traces in the given event log. However, it should still be considered a proof of concept. There is still significant room for optimization (e.g., better parameter tuning and the introduction of advanced techniques such as parallelization and caching). Additionally, the DeciClare language support will need to be expanded for future use. First, the current implementation can mine nine of the available templates (e.g., no data constraint templates). These nine are considered the most important templates (e.g., most used in the arm fracture model that was made manually in chapter 2). Support for other templates will be added in the future, although some

templates cannot be mined automatically (e.g., authorization templates), and several others require information that is not typically available in an event log. Second, the implementation only supports relation templates using the ‘at least once’ interpretation for the logical expressions of the condition and consequence sides of the constraints (e.g., if A is executed at least once, then B must be executed at least once) and the ‘at most zero times’ interpretation for the logical expression of the consequence side (e.g., if A is executed at least once, then B can be executed at most zero times). This is similar to the interpretation of the corresponding branched templates in Declare for the positive and negative versions of the relation templates. Third, DeciClare allows constraints to be not only activated by way of a decision but also deactivated. However, from experience, we know that deactivations are required much less frequently compared to activations; thus, including these in the second phase in a similar way would not be very efficient. How to add support for constraint deactivations is still an open research question and might require the introduction of a third phase. Fourth, the current implementation only supports data attributes with a finite domain. A data attribute can have a Boolean (no overlap) or categorical (allowed to overlap) domain, but continuous data attributes (e.g., temperature = 39.6°) have not yet been considered. These present a tougher challenge, as it is difficult to process their infinite domains efficiently. An extra preprocessing step by a domain expert is currently required to convert infinite domains to well-defined finite domains until support for these types of data attributes is added. Lastly, the time parameters present in many of the constraint templates are not yet supported. These were included in a previous version of the miner but were only outlined by the minimal and maximal time parameters for each rule and not mined actively. However, this subject will also be revisited as part of future research.

It is also worth noting that the current lack of a visual syntax for DeciClare does not greatly impact DeciClareMiner. When a real concrete syntax becomes available, no changes to the miner itself are required. A visual representation of the mined model can be easily created separately, as the results of DeciClareMiner are a direct instantiation of the abstract syntax of the language, which will have to be the foundation of the visual syntax anyway.

3.8. CONCLUSIONS AND DIRECTIONS FOR FUTURE RESEARCH

In this paper, we proposed a process discovery approach for loosely framed KiPs. The process discovery algorithm of DeciClareMiner can mine DeciClare process models in the form of decision-independent and decision-dependent rules. The algorithm attempts to automatically extract a complete picture of the (possibly tacit) knowledge of a process contained within a given event log.

The algorithm consists of two phases to mine both decision-independent rules and decision-dependent rules. The first phase involves a complete search algorithm inspired by the Apriori association rule mining algorithm, while the second phase involves a heuristic belonging to the class of genetic algorithms. This difference in approach is rooted in the assumption that in the context of loosely framed KiPs, the number of possible activities will be much lower than the number of combinations of available data elements on which decisions are based. This means that using a complete-search algorithm for the first phase is acceptable, while for the second phase, a heuristic is more appropriate. This assumption was confirmed in several healthcare event logs⁵. Although the concept of mining decision-dependent rules is the main contribution of this paper, the alignment of the strategies for mining decision-independent and decision-dependent rules is also a novelty. By using the waste generated in the first phase, we can turn the second phase into a much more targeted search. The process discovery algorithm was evaluated using a realistic event log of an arm fracture healthcare process and by comparing the results to a benchmark set obtained via an exhaustive search approach. DeciClareMiner considerably speeds up the discovery of a process model compared to a complete search algorithm. A trade-off was made to prioritize good results in reasonable time over a guaranteed complete model. This makes DeciClareMiner more applicable to real-life scenarios, where time and processing power are limited.

⁵ For the arm fracture event log: 9 activities and 21 data values, leading to 511 activity combinations and ~280.000 data combinations. For the Sepsis event log (see §5.2.6.1.1): 16 activities and 214 data values, leading to 65535 activity combinations and $\sim 3 \cdot 10^{64}$ data combinations.

For the emergency department event log (see §5.2.6.1.1): 116 activities and 951 data values, leading to $\sim 2 \cdot 10^{34}$ activity combinations and $\sim 2 \cdot 10^{286}$ data combinations.

In future research, we will first expand the support of DeciClareMiner for the DeciClare language. Section 3.7 reports a number of limitations regarding this support, each of which must be resolved. Additionally, a visualization layer will need to be added when a visual syntax for DeciClare becomes available. The performance of the miner with event logs containing noise will also need to be investigated. DeciClareMiner should theoretically be able to successfully handle the presence of noise in event logs, as stated in section 3.6. Nevertheless, this paper does not evaluate this property. The main questions are related to the impact of noise on the efficiency of the miner.

RESEARCH CONTRIBUTIONS OF CHAPTER 3

- The process discovery algorithm DeciClareMiner, which can automatically mine a DeciClare model from a given process event log

Chapter 4 describes EC3 and RC3 (see section 1.4).

Reference:

Mertens, S., Gailly, F., Van Sassenbroeck, D., Poels, G.: *Integrated Declarative Process and Decision Discovery of the Emergency Care Process*. (2019).

4

Integrated Declarative Process and Decision Discovery of the Emergency Care Process

4.1. INTRODUCTION

Process modeling has been successfully adopted in many different sectors. However, healthcare is one of the few sectors that have yet to fully embrace the process-driven approach (Lenz and Reichert 2007), even though some of the main concerns trending in eHealth are very similar to those of other sectors, namely, cost reduction, efficiency and patient orientation (Payton et al. 2011). Healthcare processes can be subdivided in two groups of processes: medical diagnosis/treatment processes and organizational/administrative processes (Lenz and Reichert 2007). The slow adoption of process modeling in healthcare is primarily manifested for the medical diagnosis and treatment processes. These processes typically represent the extreme end of the complexity spectrum for processes, hypercomplexity (Klein and Young 2015), and can be characterized as dynamic, multi-disciplinary, loosely framed, human-centric and knowledge-intensive processes (Mertens et al. 2017; Rebuge and Ferreira 2012). Loosely framed processes have an average to large set of possible activities that can be executed in many different sequences (Di Ciccio, Marrella, et al. 2015), leading to situations where deviations and variations are the norm rather than the exception. The knowledge workers participating in the execution of a process (i.e., doctors and other healthcare personnel) use cognitive processes to decide which activities to perform and when they are to be performed (Greenes et al. 2018). To make these decisions, they often leverage what is called tacit knowledge, meaning that they have an implicit idea of the appropriate actions to perform when certain conditions apply (Lenz and Reichert 2007). This idea might be partially based on explicit knowledge (e.g., medical or hospital guidelines), but experience usually factors in heavily.

“To make medical knowledge broadly available, medical experts need to externalize their tacit knowledge. Thus, improving healthcare processes has a lot to do with stimulating and managing the knowledge conversion processes.”

(Lenz & Reichert, 2007, p. 44)

Modeling these processes can be beneficial in many ways. A process model can serve as a means to document the process (which increases transparency for all stakeholders), to communicate changes explicitly, educate students and new process actors, to offer passive process support (i.e., using the model as a roadmap during execution) and as a foundation for active process support and monitoring (e.g., in process engines) and all sorts of management and analysis techniques (e.g., bottlenecks, process improvement...). Process modeling in healthcare has traditionally focused on clinical pathways (CPs) (a.k.a. clinical guidelines, critical pathways, integrated care pathways or care maps). Although the exact definition of what they entail can vary from paper to paper (Vanhaecht et al. 2006), the general consensus seems to be that they are best practice schedules of medical and nursing procedures for the treatment of patients with a specific pathology (e.g., breast cancer). CPs can be considered process models, but they have serious drawbacks. CPs are the result of Evidence Based Medicine, which averages global evidence gathered from exogenous populations that may not always be relevant to local circumstances (Hay et al. 2008). Therefore, CPs can be great general guidelines, but real-life situations are typically more complex. Second, comorbidities and practical limitations (e.g., resource availability) can prevent CPs from being implemented as-is. Third, the real process also comprises more than just the treatment of patients, as steps like arriving at a diagnosis and other practicalities should to be considered as well. A CP is therefore a form of high-level model of one idealized process variation. To unlock the full aforementioned benefits, a more complete view on the real processes is needed that encapsulates all possible care episodes within a certain real-life management scope (e.g., department). Of course, CPs and other medical guidelines can subsequently be used to verify whether such model of the real process follows the medical best practices as closely as possible.

Modeling healthcare processes manually can be a tedious task due to their intrinsic complexity and general need for extensive medical knowledge. Process mining techniques have been applied to healthcare processes to discover process models (semi-)automatically (Andersen and Broberg 2017; Baker et al. 2017; Basole et al. 2015; Funkner et al. 2017; Helm and Paster 2015; Huang et al. 2013,

2014, 2016; Kovalchuk et al. 2018; Orellana Garcia et al. 2015; Perer et al. 2015; Wang et al. 2017) and recent literature reviews indicate that it is still trending upwards (Ghasemi and Amyot 2016; Kurniati et al. 2016; Rojas et al. 2015, 2016). However, the existing work is almost exclusively focused on techniques that result in either imperative process models or CPs. The introduction of new process modeling languages based on the declarative paradigm (Hildebrandt et al. 2012; Pesic 2008) opens up new research avenues. Declarative languages are better suited to represent loosely framed processes, because they can model the rules describing the process variations instead of having to exhaustively enumerate all variations as imperative techniques do. The knowledge-intensive side of the processes can be supported by integrating a knowledge perspective into the declarative process modeling languages. This allows for the decision logic of the process to be modeled and makes the resulting group of languages a natural fit for loosely framed knowledge-intensive healthcare processes. Declarative process mining techniques have been applied before on healthcare processes, but not on the complete medical care process (e.g., only the treatment of already diagnosed breast cancer patients) (Maggi, Dumas, et al. 2013) and without considering the decision logic that governs the process (Burattin et al. 2015; Rovani et al. 2015).

From a methodological perspective, methodologies for the application of data mining have been developed and are widely used (Mariscal et al. 2010; Shearer 2000). However, these are too high-level and provide little guidance for process mining specific activities (van der Aalst 2011c). Therefore, methodologies were developed specifically for the process mining subdomain (van Eck et al. 2015; Rebugue and Ferreira 2012). These provide a good foundation for typical process mining applications but are geared towards imperative process mining techniques and ignore data and decision aspects of the process. The methodology of Rebugue and Ferreira (2012) is positioned to target healthcare processes, but the use of imperative process mining techniques means that they have to resort to clustering techniques in an effort to filter out some behavior and isolate smaller parts of the process. Rebugue and Ferreira (2012) also represent the methodology as a straightforward

linear approach, which is unrealistic in the context of real-life applications on loosely framed processes.

This paper reports on an Action Design Research (ADR) cycle that is part of a design science project to develop a process-oriented system that manages and supports loosely framed knowledge-intensive processes (Mertens et al. 2017, 2018; Mertens, Gailly, and Poels 2019; Mertens, Gailly, Van Sassenbroeck, et al. 2019). The research artifact of this paper is a method for process and decision discovery of loosely framed knowledge-intensive processes. This method consists of the preparation of process data to serve as input for a decision-aware declarative process mining tool called DeciClareMiner (Mertens et al. 2018), the actual use of this tool and the feedback loops that are necessary to obtain a satisfactory model of a given process. The method was iteratively improved and validated by applying it to a practical problem.

The remainder of this paper is structured as follows. The next section presents the research methodology that was applied in this paper. A brief overview of the modeling language DeciClare and the corresponding process mining tool DeciClareMiner are given in section 4.3. Section 4.4 describes the practical case and the corresponding ADR-cycle to develop a method for process and decision discovery of loosely framed knowledge-intensive processes. This is followed by a deeper analysis of the model of the emergency care process that was created as a by-product of the development of the knowledge discovery method in section 4.5. Section 4.6, we discuss the results, implications and limitations. And finally, section 4.7 concludes the paper and presents directions for future research.

4.2. RESEARCH METHODOLOGY

This paper reports on the first two stages of an Action Design Research (ADR) cycle that is part of a design science project to develop a process-oriented system that manages and supports loosely framed knowledge-intensive processes (Mertens et al. 2017, 2018; Mertens, Gailly, and Poels 2019; Mertens, Gailly, Van Sassenbroeck, et al. 2019). ADR is a specific type of design science research

for the design of research artifacts that explicitly provide theoretical contributions to the academic knowledge base, while simultaneously solving a practical problem (Sein et al. 2011). It consists of four stages:

- Stage 1: Problem formulation
- Stage 2: Building, intervention and evaluation
- Stage 3: Reflection and learning
- Stage 4: Formalization of learning

The previous section discussed the problem statement: the lack of a process mining methodology for declarative process and decision mining techniques. Therefore, the research artifact of this paper is a method for process and decision discovery of loosely framed knowledge-intensive processes. This method should consist of the preparation of process data to serve as input for a decision-aware declarative process mining tool, the actual use of this tool and the feedback loops that are necessary to obtain a satisfactory model of a given process.

For the second stage, we will do an IT-dominant ‘Building, intervention and evaluation’-stage, which is suited for innovative technological design. This stage consists of building of the IT artifact, an intervention in a target organization, and an evaluation. We started from the existing methodologies (van Eck et al. 2015; Rebuge and Ferreira 2012) to identify relevant steps in a process mining project. These activities were subsequently evaluated for their applicability to a declarative setting. As the artifact is still in preliminary development, we will limit the intervention steps to light-weight interventions in a limited organizational context (Sein et al. 2011).

The practical problem and organizational context that we will tackle is the discovery of an integrated process and decision model of the emergency care process that takes place in the emergency medicine department of a private hospital with 542 beds⁶. Data about the registration, diagnosis and treatment of patients over a span of just under two years is available in their Electronic

⁶ AZ Maria Middelaers hospital, Ghent, Belgium - <https://www.mariamiddelaers.be/>

Health Record (EHR) system. Most of this data is plain text (in Dutch), which is a typical obstacle for process and decision discovery (Sittig et al. 2008).

At the end of each iteration of this stage, the research artifact will be evaluated, and changes will be made where necessary. This evaluation will be performed by the researchers in close cooperation with the head of the emergency medicine department of the hospital. The latter will take on the role of the domain expert, who can point out problems with the application of the research artifact which, in turn, will be traced back to the research artifact itself by the researchers.

Additionally, we will perform a small-scale evaluation of the decision logic discovered in the application of the design artifact. Since this will be the first real-life application of the DeciClareMiner-tool, this is the first opportunity to verify whether the tool can fulfil its potential in real-life. This evaluation is not really part of the ADR-cycle, but rather a continuation of the evaluation of a previous design science research cycle (i.e., EC4 from chapter 3). We will select several positive constraints, as these are more interesting, with varying support from the eventual model of the emergency medicine department. The evaluation will then consist of a Likert-scale scoring of the disjunct pieces of decision logic related to each constraint by the domain expert based on medical correctness.

4.3. BACKGROUND

DeciClare (see chapter 2) is the language we used in the project for process modeling. While many currently available process modeling languages focus solely on the explicit sequencing of the activities of processes (i.e., the control flow of the process activities), DeciClare offers integrated declarative functional, control-flow, data and resource perspectives on processes. It tries to expose the decision logic that governs the sequencing of activities (e.g., activity Y must be executed because activity X was executed and data attribute ‘d’ has value 19). This decision logic is typically tacit knowledge in many loosely framed knowledge-intensive processes (Wyatt 2001), which heavily influences the information needs of the knowledge workers as well as the process execution itself

(Stefanelli 2004). By modeling or mining this decision logic, the tacit knowledge is transformed into explicit knowledge that can be used, managed, analyzed and reused.

In the previous engineering cycle of this project, we developed a tool that is able to (semi-) automatically and efficiently mine declarative process and decision logic models of loosely framed, human-centric and knowledge-intensive processes: DeciClareMiner (chapter 3). The underlying algorithm consists of two phases. In the first phase it mines decision-independent constraints (see Table 17 for explanations of the used process mining/modeling terminology) using a search strategy inspired by the Apriori association rule mining algorithm. The search strategy is complete, which means that all decision-independent constraints that fit the given parameters are guaranteed to be found. The second phase uses a heuristic belonging to the class of genetic algorithms to link seed constraints (i.e., constraints with sufficient support but insufficient confidence) with activation decisions in order to find decision-dependent constraints. This difference in approach between the two phases is rooted in the assumption that in the context of healthcare processes, the number of possible activities will be much lower than the number of combinations of available data elements on which decisions are based. This means that using a complete-search algorithm for the first phase is acceptable, while for the second phase, a heuristic is more appropriate. As a bonus, the first phase also generates all seed constraints for the second phase as a ‘waste’ product (also guaranteed), which narrows the search area of the second phase considerably.

Term	Explanation
<i>Data attribute</i>	The definition of some data property in the form of a name and a data type (e.g., the Boolean ‘hasPaid?’).
<i>Data value</i>	A specific value that a certain data attribute can have (e.g., TRUE).
<i>Declarative constraint</i>	A limitation of the allowed behavior when executing a process. This can be both in a positive (i.e., a certain activity or event is required to be executed at least a given number of times during a certain time span) and negative manner (i.e., a certain activity or event is not allowed to be executed during a certain time span). The types of constraints can be classified according to the perspective to be modeled: functional/control-flow, data and resource constraints. In this paper we will focus on existence (i.e., aforementioned positive and negative examples) and relational constraints (i.e., Response(A, B) states that if activity or event A is executed, then activity or event B must to be executed somewhere in the future before the process execution ends) as part of the functional/control-flow perspective.

<i>(De)activation decision</i>	A constraint becomes activated when the data conditions, connected with logical operators (i.e., AND, OR, NOT), of its activation decision are fulfilled by the current data context and stays active until the data conditions of its deactivation decision are fulfilled by the current data context. An active constraint is a constraint that must be satisfied for the current process execution to be valid, while a valid execution is allowed to violate an inactive constraint with no consequences.	
<i>Decision-independent constraint</i>	A constraint that has a trivial (i.e., always true) activation decision. This means that there are no data conditions that need to apply for the constraint to be active. Therefore, this kind of constraints must always be satisfied by a process execution. For example, ‘Patient registration’ must be executed at least once. Note that this term only refers to the data conditions of the activation decision. A relational constraint can have both an activation decision based on data conditions and a specific activity or event that must have occurred before the target (i.e., B in the example above) becomes required.	
<i>Decision-dependent constraint</i>	A constraint that does not have to be satisfied until the activation decision evaluates as true. It needs to be satisfied starting from when it was activated up until it (optionally) gets deactivated when the deactivation decision evaluates as true. For example, the constraint ‘after an examination by a doctor, surgery must be performed at least once’ and an activation decision that evaluates as true when it is a trauma case and when the patient exhibits signs of shock and/or heavy local inflammation or pain during the examination.	
<i>Conforming activation</i>	A constraint that is activated and satisfied by a certain (partial) trace (e.g., consequence side of an activated relation constraint is satisfied).	
<i>Violating activation</i>	A constraint that is activated and violated by a certain (partial) trace (e.g., consequence side of an activated relation constraint is not satisfied).	
<i>Conformance percentage (a.k.a. confidence) of a constraint</i>	The number of traces in which the constraint is activated and satisfied divided by the total number of traces in which it is activated.	
<i>Support percentage of a constraint</i>	The percentage of traces in which the constraint is activated (independent of whether it is a satisfactory or violating activation).	
<i>Model fit</i>	How well a model fits the set of given observations. This typically directly correlates to its predictive power for future observations.
<i>Overfit</i>	A mined process model that corresponds too closely or exactly to the data used for the mining step and may therefore fail to fit additional data or predict future observations reliably.
<i>Underfit</i>	A mined process model that corresponds too little to the data used for the mining step and may therefore fail to encapsulate the logic of the underlying process with enough details to predict future observations reliably.

Table 17. Explanations of some process mining and/or declarative modeling terminology

4.4. DISCOVERING THE EMERGENCY CARE PROCESS WITH DECICLAREMINER

Fig. 18. A high-level overview of the method for process and decision discovery of loosely framed knowledge-intensive processes

Fig. 18 presents a high-level overview of the method that was the result of the ADR cycle described in this paper. It shows how raw data is extracted from the database of the Information System (IS) of an organization and transformed into a satisfactory process and decision model. The internal structure or language of the inputs and outputs (grey) as well as the entity responsible for the execution (blue) of each step have also been added. The role of consultant in the practical application of the method was performed by the first author as is typical in ADR.

The remainder of this section describes what each step entails, what lessons were learned and how it was performed in the practical application.

4.4.1. Extraction

The IS of an organization contains the data needed to execute all processes in the organization. Typically, it contains data related to many different processes. The IT department must first identify the data related to the target process and extract it from the database to some sort of temporary data structure.

In the practical application, the hospital IT department provided us with an export of six database tables from their internally developed EHR system spanning the period from 31/12/2015 until 22/09/2017. The data was exported from the original relational database management system as CSV-files. An overview of the tables, the relevant data fields and the relations (translated to English) is given in Fig. 19. The admission time of a patient is stored first, while a possible discharge time is added to the same record afterwards. Transfers are stored in a separate record in the same table. During a triage, the nurse can assign a general category (i.e., manchesterFiche), a subcategory (i.e.,

Fig. 19. An overview of the original database tables (relationships in green and identified activities in blue)

manchesterQuestion) and a priority color coding to the patient, which are stored in the ManchesterTriage table. Diagnostic imaging tests (i.e., RX, Echo, CT and NMR) are all stored in the RX table and the description containing the exact type of test. It is worth noting that we only had access to the date on which the scan was made, not the time. The doctor-patient interactions are contained in the ContactLine and ContactData tables. Each ContactLine -record represents an update to the patient file in the context of a specific contact (i.e., the interactions of one specific doctor with the patient) and is connected to one or more ContactData -records. A ContactData -record contains the name and updated value of a textbox in the user-interface of the EHR (e.g., physical examination). The 'lastUpdate'-field of ContactLine represents the date and time when the doctor registered the activity in the EHR, not when the actual activity of the contact was performed.

4.4.2. Data preprocessing

The use of a certain database management system imposes certain structural limitations on how data can be stored in the database. These limitations are not relevant for the subsequent steps of the proposed method, and therefore, a conceptual model needs to be constructed that makes abstraction of any structural limitations of the original database and other implementation-related aspects.

Fig. 20. UML class diagram of the conceptual model for the preprocessed data

Fig. 20 represents the conceptual model that was reconstructed from the raw EHR data. A Patient -class was omitted as our data set was anonymized, and therefore, no general patient-related data was included (e.g., gender, age, etc.). Without this restriction, we would advise the inclusion of a Patient -class that is connected to the CareEpisode -class through a one-to-many relation. The Admittance_Transfer_Discharge -table was split into three classes: CareEpisode, Transfer and ExternalTransfer. A care episode represents a single admittance of a patient that can end with a discharge or a transfer to another department. An object of the Transfer -class represents an internal transfer (i.e., within the ER) and an object of the ExternalTransfer -class, which is defined as a subclass of the Transfer -class (i.e., it inherits its attributes and is treated in the same way), represents a transfer to another department. Furthermore, the Triage-, Imaging- and Contact- classes are also linked to the CareEpisode -class. The RX- table was transformed to the Imaging -class. The description field of a record in the RX -table was used to extract a more specific RX -activity, stored in the ImagingTechnique -enum, and RX -records containing multiple imaging techniques in their description were split into separate Imaging -objects (e.g., RX description = “Echo and CT of right foot” is transformed to separate Imaging-objects, one with the echo imaging technique and one with the CT imaging technique). And finally, the ContactLine- and ContactData -tables were transformed to the Contact- and ContactLine -classes. The former represents the general contact and contains

most of the data stored in the ContactLine -table, while the latter represents an inserted or updated contact line and contains the insert or update date/time as recorded by the lastUpdate -field from corresponding record in the ContactLine -table and the data from the matching record in the ContactData -table.

A typical difficulty in projects such as this one is that much of the data is stored as plain text in the database (e.g., the text -attribute in the ContactData -table). This data cannot be used without introducing some structure. Although more advanced text mining and natural language processing techniques exist, for example TiMBL (Daelemans et al. 2009), we applied the more rudimentary but also more straightforward set of techniques described in Fig. 21. Of course, more advanced techniques can be incorporated without changing the proposed method.

The bags-of-words models of the plain text data fields (1) removed the grammar and word order but retained the multiplicities of individual words (Harris 1954). The subsequent application of lemmatization (2) reduced each word to its lemma (i.e., dictionary form). Stemming is a simpler alternative, though this leads to more information loss. In the next step, the synonyms, nominalizations, adjectivizations and abbreviations were each consistently replaced by a preferred term and common typos by the intended words (3). Because a medical jargon dictionary/thesaurus like SNOMED CT⁷ is not yet available for the Dutch language, we manually defined lists of words for this purpose, starting with the most frequent words. The resulting multisets of words for each data field were then filtered to remove stop words and words with no medical relevance (4). Again, a manually composed list of medically irrelevant and stop words was used. At this point, the plain text data fields had been transformed to multisets of normalized keywords. However, using them directly would lead to a huge number of keywords that do not consider the anatomical or pathological hierarchy ranging from specific low-level terminology to more general high-level terminology. Instead, we chose to replace some rather specific words by more general terms (5), to reduce the number of potential keywords, and in some cases annotated words with closely related

⁷ <http://www.snomed.org/>

Fig. 21. Techniques to structure plain text data fields with example

general terms (6), in an effort to reach the desired granularity level. The generalization and annotation steps in this project were also performed using manually composed word lists (SNOMED CT hopefully will provide a more complete hierarchy for the Dutch language soon). The granularity level is one thing; but we also wanted to include a mechanism to control the quality and quantity of unique keywords for each data field. So finally, we manually selected the relevant terms as keywords from lists that were made of the most frequent words per data field (7).

4.4.3. Log preparation

To mine a process model, the traces of an event log describing the data need to be reconstructed. This means walking through the data of each historic execution of the process, as structured by the conceptual model (Fig. 23).

Fig. 23. The intermediate activities to go from the preprocessed data to an event log

Therefore, in the next step we reconstructed the chronology and created a trace for each care episode. The reconstruction of the activity events is illustrated with an example in Fig. 22. The activity events are shown in blue (corresponding data events are omitted), with green arrows showing the data on which it is based). Each trace starts at time 0 and the relative sequencing is done by using the differences (in seconds) between the timestamps. The relative timestamps were used because the timestamps in the data were randomly shifted on a per-patient basis due to privacy

Fig. 22. An example of how the internal representation (preprocessed data) is transformed to a trace

concerns, and therefore, any information (e.g., number of simultaneous treatments) that they contained was already lost.

The relevant data values are transformed to data events in the trace (e.g., data value ‘bedNr’ is either a number or NULL and gets transformed to a Boolean data event ‘Patient needed bed?’ that occurs during the triage activity event). The most interesting data to reconstruct the decision logic of the process is stored in the plain text attributes (e.g., the attribute ‘text’ in the ContactLine -class), which have been preprocessed to multisets of keywords as described in the previous subsection. For each of those attributes, the keywords were transformed to categorical data events (e.g., ‘Description of Medical history’ = ‘foot’). Practically, we used the XES-format for the event log and used a custom extension to include general data events (i.e., adds data events with a timestamp that do not need to be directly linked to an activity event).

The imaging activities have a date, but they lack a timestamp. As a workaround, they were given a chronological position in the trace with a custom matching algorithm that matches its preprocessed description with the preprocessed text of the other activities and then positions the resulting imaging activity according to a set of rules determined by the domain expert. For example, the RX-activity in the example of Fig. 22 is positioned between the ‘Physical exam’- and the Conclusion-activity. The rules state that the RX must have been requested before the doctor could have made its conclusion about the patient. However, the preprocessed description of the RX matches with none of the preprocessed text of the previous activities. Therefore, the RX-activity is placed directly after those other activities.

Several iterations were needed to create a satisfactory event log. Yet, the need for these iterations only can come up during later steps, because a domain expert must make this assessment. It would be possible to add an intermediate evaluation of the constructed event log at the end of this step, depending on the urgency of the project and the availability of the domain expert. However, the next step (mining) is automated anyway. Therefore, we decided against adding the extra evaluation step. We just used the created event log to mine a model and had the domain expert evaluate the event

log and the mined model simultaneously during the normal evaluation step. The first iterations of the event log creation focused on the identification and granularity of activities and data elements, while the sequencing of activities with missing timestamps (i.e., imaging activities and lab tests) also required multiple iterations. Each iteration incrementally improved the event log to better reflect reality. The resulting event log contains 41657 traces using 116 activities and 951 data values. Some additional statistics about the log are presented in Table 18.

	Total of all traces	Minimum per trace	Maximum per trace	Average per trace	Median per trace
Duration (hours)	133686	0.067	22.3	3.2	2.9
#Activity events	625682	3	128	15	17
#Data events	1350331	2	166	32.4	32

Table 18. Additional statistics on the event log

4.4.4. Mining a process model

Fig. 24. The intermediate activities to go from the event log to an integrated process and decision model

Process mining tools can use the event log composed in the previous steps as input for the discovery of process models (Fig. 24). Process mining techniques that create imperative process models often resort to the use of slice and dice, variance-based filtering (e.g., clustering) or compliance-based filtering to reduce the complexity of a given event log (van Eck et al. 2015; Rebuge and Ferreira 2012). For process mining techniques that create declarative process models this is not a necessity, as these can handle more of the complexity, but can also optionally be used to focus on a specific part of the dataset. While the other activities can be performed automatically, the consultant responsible for the project should have decided on the branching and minimal confidence and support levels (see Table 17 for definitions) for DeciClareMiner to mine models with by now. These can be used to control the level of detail and complexity of the resulting models in order to prevent overfitting and manage the impact of noise in the event log.

DeciClareMiner was executed with several different parameter settings on an Intel Xeon Gold 6140 for the emergency care event log. The parameter settings varied the minimal support level of the mined constraints from 0.001% to 0.005% to reduce the overfitting effects by forcing the miner to generalize the activation conditions to apply to at least multiple traces. The minimal conformance level was always set at 100%, which means that only behavior that is never violated in the event log can be found. Although the principal technique can handle noise when considering mined constraints in isolation, lower conformance levels can result in inconsistent models (Di Ciccio, Mecella, et al. 2015) and tools to resolve this (Di Ciccio et al. 2017) are not yet available for DeciClare. The branching level for the general mining experiments was 1, so no branching, but constraints with higher branching (up to 58) were separately explored to answer specific questions (e.g., what activities are executed last?). The second phase of DeciClareMiner was executed multiple times with the same parameter settings, because this phase is not deterministic, and these results were combined at the end.

Postprocessing is applied automatically to the raw output of DeciClareMiner. This eliminates equivalent decision rules (i.e., combinations of data values that uniquely identify the same cases) and returns a much smaller model with the given support and conformance levels. However, it is not necessarily true that when two decision rules are equivalent in the given event log, that this is also true in real life. Due to the incompleteness of the log (i.e., not every possible case occurring) it can be impossible to distinguish between the real-life decision rule and other decision rules that identify the same set of cases by coincidence (e.g., when the patient got a red color code and a remark mentions the neck area during the triage, at least one NMR will be taken afterwards, but the same patients might also be identified uniquely with the red color code and a remark mentioning sweatiness just because this limited set of patients happened to coincidentally have been sweaty as well as have had neck issues). Therefore, the final result⁸ consists of two postprocessed models with

⁸ <https://github.com/stevmert/discoveryMethod>

different support levels and a separate repository of all the mined decision logic from before postprocessing.

4.4.5. Evaluation

After each mining iteration, the correctness of the reconstructed event log and the mined DeciClare-model needs to be validated by a domain expert (see Fig. 18). These process models are difficult to validate directly because there is nothing to compare them with. The understandability issues associated with declarative process models (Haisjackl et al. 2014) and sheer number of constraints in the models also made it infeasible to go over them one-by-one. Instead, we adopted a similar approach to Braga et al. (2010) and Guizzardi et al. (2013), which uses the visual simulation capabilities of Alloy Analyzer to evaluate a conceptual model. We advise the use of the DeciClareEngine model simulator (see section 5.1.5.2) to demonstrate what the mined process model knows about what can, must and cannot happen during the execution of specific traces (see Fig. 25). DeciClareEngine shows what activities have been executed at that point in the trace (top), the corresponding data events (middle) and the activities that can be executed next according to the process model (bottom). The user can view the reasons why a certain activity must/cannot be executed as well as any future restrictions that might apply (in the form of constraints) at any time by clicking on the ‘Explanation’-, ‘Relevant Model’- and ‘Current Restrictions’-buttons.

For the evaluation of the model of the emergency care process, we also created a module for DeciClareEngine that allows the user to replay the traces of the event log side-by-side with the original data of the EHR (see Fig. 26). This module has an interface that gives an overview of the activities of the trace (left), and one click on an activity sets the state of the simulator to just after the execution of that activity. The original EHR data can be opened by clicking on the ‘Report’-button next to the activity (right).

The domain expert was asked to give feedback during each step of the simulation of a number of randomly-selected patients. Before the execution of the next activity, he was asked the questions from Table 19. This information was used to get an idea of the correctness of the event log and

process model and the value of the model to the doctors and the hospital in general. This was repeated each time a new process model was mined. A collection of stopping criteria, for both the reconstructed event log and mined model, were used that all needed to be satisfied:

1. There should be no discrepancies between the original data timeline and the reconstructed sequence of activities of the simulated patients (i.e., in the correct order or at least in a realistic order when the original data does not specify an exact time).
2. All data events of the simulated patients should be based on data that was actually available at that time in the process (i.e., no data events using data that was added in retrospect or at an unknown time).
3. The mined model needs to allow and incapsulate all observed behavior, while restricting unobserved behavior as much as possible. Overfitting rules are the result of unobserved but acceptable behavior, which is generally undesirable. However, we envision that for real-life usage the process engine will be used to automatically relax or prune away these overfitting rules (i.e., self-adaption) when previously unobserved behavior was observed due to a user overruling the model. This also reduces the need for a full validation of the mined model as it will be validated gradually by the users, with each overfitting rule serving as a warning that the deviating behavior is not typical.

DeciClareEngine - SplitV1ResPP_resultRules_100.0_0.001.data (SplitV1DataModel)

Current time: 4132

Current resource: null

Executed Activities:

- Aanmelden(0,2)
- Triage(1614,1616)
- Transfer naar C005(1620,1622)
- Transfer naar C005(2040,2042)
- Nieuw contact aanmaken(2674,2676)
- Labo aanvragen(4129,4131)

Data Events:

- 1: Zvnm zvm? = 1 (+)
- 1615: Is slachtoffer van ramp? = false (+)
- 1615: Patient heeft bed nodig? = true (+)
- 1615: Triage op spoed? = true (+)
- 1615: Triage kleur? = GROEN (+)
- 1615: Triage categorie? = buikstreek (+)
- 1615: Triage categorie? = blaas (+)
- 1615: Triage categorie? = abdomen (+)
- 1615: Triage categorie? = urinair (+)
- 1621: Patient locatie? = c005 (+)
- 2675: Omschrijving contactijn? = pijn (+)
- 2675: Omschrijving contactijn? = buikstreek (+)
- 2675: Omschrijving contactijn? = blaas (+)
- 2675: Discipline contactijn? = spoed (+)

No Resource Events

tierA_satisfaction		
tierB_possibleActivity Satisfaction		
CT aanvragen(3,5)	Explanation	Relevant Model
Echo aanvragen(3,5)	Explanation	Relevant Model
RX aanvragen(3,5)	Explanation	Relevant Model
RX van onbekend type aanvragen(3,5)	Explanation	Relevant Model
Anamnese(5704,5706)	Explanation	Relevant Model
Labo aanvragen(5704,5706)	Explanation	Relevant Model
Technisch onderzoek beschrijven(5704,5706)	Explanation	Relevant Model
Verloop beschrijven(5704,5706)	Explanation	Relevant Model
tierC_resourceViolation		
tierD_timeViolation		
tierE_violation		
Administratie beschrijven(0,2)	Explanation	
Anale manometrie beschrijven(0,2)	Explanation	
Anatomopathologisch onderzoek beschrijven(0,2)	Explanation	
Antibiotherapie beschrijven(0,2)	Explanation	
BERA beschrijven(0,2)	Explanation	
Beleid beschrijven(0,2)	Explanation	

Relevant Model Current Restrictions

Fig. 25. The interface of the DeciClareEngine model simulator

Fig. 26. The log replay console that contains the original EHR data for the simulated patient

Question	Purpose
Based on the (EHR) data, are there any activities that might be missing at this point?	Identify missing activities
Is the order of activities how you would expect based on the (EHR) data?	Identify reconstruction mistakes
Does the model allow activities to be executed next that should not be allowed?	Detecting underfit in model
Does the model prohibit activities to be executed next that should be allowed?	Detecting overfit in model
Does the decision logic for why certain activities are allowed/prohibited reflect real decision-making processes?	Measuring knowledge extraction of model

Table 19. The questions for the domain expert and the underlying purpose behind each of them for the validation

4.5. EXTRACTING KNOWLEDGE BY MODELING THE EMERGENCY CARE PROCESS

The goal of the ADR-cycle described in the previous section was the development of a method for process and decision discovery of loosely framed knowledge-intensive processes, however, the cycle also resulted in the creation of the integrated process and decision model of the emergency care process as a by-product. The most complete DeciClare-model (support level 0.001%) that was discovered of the emergency care process contains 23411 decision-independent and 16609 decision-dependent constraints. Fig. 27 presents a simplification of the decision-independent constraints of

Fig. 27. Simplified Declare representation of the mined process model

the model using the Declare visual syntax (Pesic 2008). It shows only the activities done by emergency doctors (so no specialist activities), combines the external transfers that do not have a specific relation constraint into a generalized activity called ‘External transfer (other)’, omits chain constraints and omits other constraints with a support lower than 0.25%.

This model provides a rare look into the diagnosis and treatment of patients in an emergency medicine department. The broad outlines of the model will not surprise any of the process actors, as they already had an implicit idea of the general way things are done at the department, but the model makes them explicit so that it becomes accessible to outsiders (i.e., management) and opens the door to further analysis and optimization. However, things get more interesting when diving deeper into the model. The goal of modeling the emergency care process was to transform tacit knowledge possessed by the knowledge workers into explicit knowledge. The tacit knowledge captured by the model consists of two main components: the functional and control-flow perspectives describing the occurrence and sequencing rules concerning the activities and the decision logic that determines the context in which each occurrence and sequencing rule applies. During the validation step, the focus is mostly on the former (i.e., does the model allow/prohibit the expected activities at each

decision point?). The latter was validated in the context of specific patients, not on a per constraint basis. The medical personnel typically leverage their knowledge and experience when deciding on how to deal with a specific patient. This tacit knowledge can apply to all patients (e.g., registration always first) or to a specific subset of patients based on the characteristics of the patient and other context variables (e.g., if shoulder pain and sweaty, take echo of the heart). It would be valuable to evaluate the full decision logic that was mined as the activation decision of a specific constraint, because it contains all possible contexts that were found to lead to the activation of the constraint. Do these contexts match the tacit medical knowledge of the doctors?

Therefore, we selected four activities that appear at least once in a fraction of the available traces, ranging from relatively frequent to almost never (see Table 20). The decision logic that was mined for these rules was able to correctly identify many traces in which the activity would need to be executed before the actual execution. We listed each disjunct part of the mined decision logic and asked the domain expert to score each part on a Likert-scale ranging from medical nonsense to realistic decision logic. Note that requesting an RX-scan is generally considered standard procedure, therefore there is often not an identifiable reason for requesting it. This is reflected by the numbers in Table 20: only 11% of the traces in which an RX-scan was requested could be identified uniquely (i.e., the mined decision logic activates the constraint stating that an RX-scan needs to be requested for 1603 out of the 14640 traces in which an RX-scan was requested), independent of the realism of the mined decision logic. The model still allows an RX-scan to be requested in the remaining 89% of traces in which it was eventually requested, it is just that no consistent reason was discovered to make it mandatory.

The results are summarized in Fig. 28. The overall trend is clear: the more cases to learn from, the more realistic the mined decision logic of the activation decisions. For the most frequent of the evaluated constraints, requesting at least one RX-scan, 16.3% of the decision logic was deemed realistic decision logic and another 43.5% was close but lacked minor details. For requesting a CT-scan, the second most frequent of the evaluated constraints, these numbers already drop to 4.7% and

	#Traces at least once (% of traces in event log)	#Traces uniquely identified (% of traces)	Number of decision rules verified by the domain expert
Request an RX-scan	14640 (35.14%)	1603 (10.95%)	184
Request a CT-scan	5512 (13.23%)	1092 (19.81%)	127
Transfer patient to D503 (Gastroenterology and Endocrinology)	1565 (3.76%)	735 (46.96%)	355
Request an NMR-scan	140 (0.34%)	52 (37.14%)	98

Table 20. The four activities used for the evaluation

17.3%, respectively. For the least frequent constraints, transfer to D503 and requesting an NMR-scan, none of the mined decision logic was deemed truly realistic and less than 3% was close with just some minor details lacking.

When discussing these results with the domain expert, some interesting observations came up. For one, the selection of the constraints for which the decision logic would be evaluated was not ideal. We (=the academic researchers, not the domain expert) decided to focus on positive constraints (i.e., requiring the execution of an activity), because the decision logic of negative constraints (i.e., prohibiting the execution of an activity) that we came across during the validation steps seemed to be mostly obvious statements (e.g., in the context of a patient with dementia, a transfer to the prenatal department was prohibited). Hence, we assumed that the positive constraints

Fig. 28. The results of the evaluation of the mined decision logic by the domain expert

would potentially contain the most interesting decision logic. However, if the required activity is standard procedure, then the positive constraint will not have much meaningful decision logic to be discovered, but rather, the negative counterpart might have more meaningful decision logic to be discovered to identify the contexts in which the doctors divert from the standard procedure. Secondly, we selected the activities related to scans because the different scans had very distinct frequencies of occurrence in the event log, while the transfer to D503 was chosen because its frequency fell nicely in between the large gap of frequencies of the CT- and NMR-scans. However, the domain expert noted that the NMR-scan are practically never requested by an emergency doctor, but rather by a doctor from another department who already requests it while the patient is still waiting to be transferred to that department. So, in a way, this activity is not really part of the emergency medicine process. The transfer to D503 on the other hand is apparently a broad scoped department. Patients with all sorts of internal issues are send there. Hence, the reasons for needing a transfer to that department can vary widely, making the mining task much harder than for some other departments. And finally, we also discussed the reasons why certain chunks of decision logic might not have been found:

- A) Non-determinism and time limitations: the technique is not deterministic and must search an enormous search space in limited time, so it is possible a certain decision was just not found in the available search time.
- B) Data quality issues: some data values were not written down in the patient's file by the doctors or not identified as a relevant keyword during the event log creation. The latter can be due to the fact that the techniques used to preprocess the natural language data and subsequently transform them to categorical data values are rudimentary. For example, the categorical data value 'fever' can be the result of both describing that the patient has a fever or has no fever. No contextual information was used to annotate these keywords. Applying more advanced natural language processing techniques has the potential to greatly improve the amount of tacit knowledge that can be transformed to explicit knowledge. Additionally, the

anonymization of the dataset meant that the medical history of the patient, the age and even the gender of the patient was not available. This is often crucial information to reconstruct the decision logic. For example, the model allows for almost every patient to be send to obstetrics just after registration, because it cannot differentiate between genders and age ranges for which a pregnancy is generally relevant or applicable. Of course, adding this information to the event log would result in a more detailed model of the process. However, without this information, it can be impossible to discover the corresponding chunk of decision logic, or at least discover the correct decision logic fully.

- C) Data co-occurrence: multiple sets of data values can occur simultaneously in the same contexts. The postprocessing step of the discovery algorithm picks just one of these sets (i.e., the one that occurs the most frequently or just the first one in the list if they occur with the same frequency). However, it is possible that one of the sets of data values that was actually better at describing the underlying medical knowledge is discarded this way, due to it applying to the context of the same or a mere subset of cases to which the selected set applies. Improving the data quality will reduce the impact of this issue. To eliminate it, a full review of the mined decision logic by medical experts would be necessary, however, that would be a very time-consuming step that somewhat eliminates the benefits of an otherwise largely automated technique.
- D) Inconsistent application: if a certain tacit rule was applied inconsistently, this serves to muddy the waters. We configured the discovery technique to only search for rules that seem to be followed consistently in the corresponding context(s). This means that a bit of noise, like a one-time misapplication of this rule, will lead to that rule not being discovered.

During this case study, several ideas and opportunities came up to increase the success rate of similar projects in the future at the emergency medicine department of AZMM. The benefits of storing additional process-related information became more concrete, which further increases the willingness to make changes to the current situation. Some ideas entail changes to the underlying

information system. For example, storing more timestamps concerning radiology and lab activities (i.e., time of request, time of execution, time when the results were made available, time when the doctor looks at the results...). While other ideas involve to more radical changes to the way that the medical personnel go about their business. A medical scribe was made available during a test period to relief the doctors of most of the writing activities in the EHR and at the same time to make the input more uniform. An expansion of the current capabilities to generate default texts based on shortcuts is also under consideration, possibly in combination with all options/answers so that the doctor just needs to remove what is not applicable instead of having to type it. An even more ambitious idea of adding some sort of spell correct to interface of the EHR was also discussed. All this extra structure in the textual input of the EHR will result in more precise data in the future, and consequently, more precise process models.

4.6. DISCUSSION

Fig. 29. The imperative process model as generated by the Disco-tool

The Disco-tool⁹ was used to create an imperative process model that contains all activities, but ignores the data events (not supported), and shows just the 1.1% most frequently travelled paths (see Fig. 29). This is an example of how traditional process discovery techniques would represent the emergency care process of the AZ Maria Middelaers hospital. Despite the omission of data events and 98.9% of the less frequently travelled paths, the result is still an unreadable model that is often

⁹ <https://fluxicon.com/disco/>

referred to as a spaghetti model (van der Aalst 2011a). This is caused by the 28980 unique variations in the log (out of 41657 patients), as identified by Disco when only considering the activity events, of which the most frequent variation occurs just 686 times (1.65%). It is a typical problem when modeling healthcare processes that results in poor usability of the resulting models (Fernandez-Llatas et al. 2015). This further strengthens the claim that imperative modeling techniques are not suitable for loosely framed processes (van der Aalst 2013; van der Aalst et al. 2009; Goedertier et al. 2015; Mertens et al. 2017; Rovani et al. 2015), which includes many healthcare processes. Therefore, several declarative modeling languages were developed to better cater to the needs of these kinds of processes (Goedertier et al. 2015). Our focus on loosely framed knowledge-intensive healthcare processes lead to the development of DeciClare. DeciClare incorporates generalizations of many aspects of the other declarative modeling languages, but additionally includes an integrated data perspective. This enables the inclusion of context-based knowledge, like decisions, in the process models.

The process and decision model certainly revealed some useful insight into the emergency care process of the AZ Maria Middelaers hospital. Despite the rudimentary data analysis and other limitations, it managed to transform a lot of tacit knowledge into valuable and explicit knowledge. Of course, much of the discovered knowledge is somewhat obvious to the experienced doctors in the department, but now it is also readily available for less experienced doctors and other stakeholders that want to gain some insight into the internal workings of an emergency medicine department. Having the explicit rules for the occurrence and sequencing of activities and the corresponding decision logic available enables deeper analysis and optimization of the working habits of the emergency doctors. The deeper evaluation of the decision logic in the previous subsection did show that the technique was able to expose some real decision logic automatically, even for those more difficult positive constraints. This first application of the proposed technique shows the potential that it can have to make previously tacit knowledge explicitly available for all

stakeholders. Process and decision models, like the one mined in this paper, have the potential to unlock a new tier of applications further along the line. For example:

- An explicit justification as to why certain activities are performed in specific cases and others are not.
- The decision logic of treatments in the model can be compared to the available clinical guidelines. Do the doctors follow the guidelines? If not, what are the characteristic and decision criteria of the cases in which they diverge from the guidelines? Are these justifiable?
- The decision logic of different doctors can be compared. Do the doctors treat the similar patients similarly? What are the differences? Can adjustments be made to make the service level more uniform?
- The knowledge contained by models will make simulations much more realistic and flexible. For example, how would a department address an increase in a group of pathologies by 5%? Would this create new bottlenecks? Can these be eliminated by changing, if possible, some decision criteria?
- The knowledge made explicit by the mined models can be used to bring interns more quickly up to speed or even to educate future nurses/doctors as part of the education system.
- The mined models can be used for evidence farming (Hay et al. 2008). This is the practice of posteriori analysis of clinical data to find new insights without setting up case-control studies. Evidence farming can be used as an alternative or, even better, as a supplement to evidence-based medicine.

During the application of the developed method, we encountered several typical limitations of process discovery projects. Different types of noise in the data that was used to create the model are the primary limitation. This is real-life data that was recorded without this sort of applications in mind, so it would be unrealistic to expect perfect data. The main forms of noise (van der Spoel et al. 2013) that were encountered are the following:

- Sequence noise: errors in, or uncertainty about, the order of events in an event trace.

In this project we did not have an exact timestamp of the radiology and lab activities. So, a workaround was devised that is by no means perfect. Additionally, the timestamps that we did have are those of when the doctors or nurses describe activities in the EHR, but this does not necessarily reflect the timing and order of the actual activities performed. Some doctors prefer to write in the EHR in batches. Consequently, the preservation of the sequence information is dependent on whether they write it down in the same order as they performed the activities (which was generally the case). Although there is no way to fully resolve this, aside from a whole new system to record the data, we did use names for the activities that correctly reflect the data that we had. Many activities were named like ‘Describe ...’ or ‘Request ...’ to signify that we cannot make a statement about when the actual activity occurred, just when the description or request was made.

- Duration noise: noise arising from missing or wrong timestamps for activities and variable duration between activities.

The issues described for sequence noise also apply here. We only had timestamps of when an update was saved to the EHR. Hence, there was no timestamp of the actual activity or even a duration. We gave every activity an arbitrary duration of two seconds for practical reasons, but this does not reflect reality.

- Human noise: noise from human errors such as activities in a care path which were the result of a wrong or faulty diagnosis or from a faulty execution of a treatment or procedure.

When writing in the EHR, doctors and nurses will unavoidably make mistakes: typos, missing text, missing activity descriptions, activity descriptions entered in the incorrectly activity text field... But even correct description of activities can cause problems as a different jargon will make it more difficult to detect patterns. We preprocessed the plain text data to resolve the more frequent typos and used a list of synonyms and generalizations to deal with the use of different jargons. However, these are not exhaustive, and thus, could still be improved. The

release of an official dictionary and thesaurus of medical jargon in the Dutch language would be a big step in the right direction.

In terms of the event log imperfection patterns as described by Suriadi et al. (2017), we encountered the following patterns: form-based event capture, unanchored events, scattered events, elusive cases and polluted labels.

Other specific limiting factors were already discussed in section 4.5 (see A, B C and D). As with most process mining projects, the assumption was also made that the event log is complete (Ghasemi and Amyot 2016). This means that we assume that every possible type of patient, diagnosis, complication, treatment... that can occur also occurred in the event log. Of course, this is not realistic. The result is that the process models will generally overfit the event log (i.e., it was never observed that a patient needed more than 4 CT-scans, so the model says this cannot occur, while there might be infrequent cases for which this could be possible). This can be resolved slowly by gathering more and more data, pruning away or relaxing the overfitting constraints.

The issues described above lead to a loss of accuracy in the resulting process and decision model. Some knowledge is lost, while other knowledge that was learned does not correctly reflect reality. Furthermore, the model only considers the emergency medicine department of one hospital. The generalizability of the model is therefore low. Every hospital and/or department has their own capabilities, limitations and way of doing things.

4.7. CONCLUSION AND FUTURE RESEARCH

This paper describes an ADR-cycle to develop a method for process and decision discovery, centered around the DeciClareMiner tool. The method describes the steps to go from raw data in the IT-system of an organization to a detailed process and decision model of a loosely framed knowledge-intensive process. The use of a declarative process mining technique with integrated decision discovery is novel and requires modified approach compared to existing methodologies for process mining based on imperative process mining techniques. As part of the ADR-cycle,

DeciClareMiner was applied for the first time to a real-life loosely framed knowledge-intensive healthcare process that takes place in the emergency medicine department of a private hospital with 542 beds¹⁰. The emergency medicine department is home to one of (if not the) most diverse, multi-disciplinary and dynamic processes in a hospital. Consequently, process and decision discovery in this setting is very challenging. Most of the process mining papers in a healthcare setting focus on a limited number of steps, aspects or pathologies (e.g., clinical pathways) to reduce the complexity of the task at hand. In contrast, the scope of this project was set out to be as comprehensive as possible with the available data. All recorded activities performed as part of the process in the emergency medicine department as well as all pathologies were considered from the viewpoint of the functional, control-flow and data perspectives of the process without resorting to filtering or clustering techniques to reduce the complexity.

The ‘spaghetti’ model that resulted from applying an imperative process mining technique to the data (Fig. 29) further strengthens the claim that imperative process modeling is not suitable for these types of processes. The understandability advantages (for human users) of imperative over declarative languages is completely negated by the high flexibility needs of the process and the resulting enumeration of all possible paths. Declarative languages are better suited due to their implicit incorporation of process variations. The combination of this declarative viewpoint and an extensive data perspective enables DeciClare models to capture both the loosely framed and the knowledge-intensive characteristics of the targeted processes. As a result, the discovered DeciClare model of the emergency care process offers a first glimpse into the previously tacit knowledge about how patients are diagnosed and treated at the emergency medicine department of the hospital. Transforming this tacit knowledge to explicit knowledge can have many benefits to the department, the hospital and even to the medical field in general (e.g., transparency, analysis, comparison, optimization, education...). The evaluation demonstrates that realistic functional, control-flow and decision logic can be identified with the developed method.

¹⁰ AZ Maria Middelaers hospital, Ghent, Belgium - <https://www.mariamiddelaers.be/>

As future research, we need to make a distinction between the developed method and the practical case. In the context of the method, we will investigate how more advanced text mining and natural language processing techniques, like TiMBL, can be integrated into the method directly as an optional activity in the ‘Log preparation’-step. From the perspective of the practical case, a project like this does not really have a definite ending. It will need to be repeated continuously as initiatives to improve the data quality start to bear fruit. The quality of the discovered process and decision model will improve with more and better data, which in turn externalizes more and more real-life medical knowledge.

RESEARCH CONTRIBUTIONS OF CHAPTER 4

- A method for declarative process and decision discovery, describing the steps starting from the data extraction up to the discovery of a satisfactory model of a given process

Chapter 5 consists of two papers. Section 5.1 describes EC5 and section 5.2 describes EC6, RC4, RC5 and RC6 (see section 1.4).

References:

§5.1: Mertens, S., Gailly, F., and Poels, G.: *A Generic Framework for Flexible and Data-Aware Business Process Engines*. In: H. Proper and J. Stirna (eds.) FAISE Workshop at CAiSE'19. LNBP, vol. 349, pp. 201–213, Springer, Cham (2019).

§5.2: Mertens, S., Gailly, F., Van Sassenbroeck, D., Poels, G.: *Comparing strategies to generate experience-based clinical process recommendations that leverage similarity to historic data*. In: 2019 IEEE International Conference on Healthcare Informatics (ICHI). IEEE (2019).

5

Operational Support for Loosely Framed Knowledge- intensive Processes

This chapter consists of two related papers. The section 5.1 describes development of a data-aware BPE that supports the execution of a declarative process model. The resulting BPE, called DeciClareEngine, can execute a given DeciClare model by categorizing every activity according to its availability for execution as the next activity at any time during the execution of a process instance. However, loosely framed KiPs typically have a number of activities that can be executed next, presented as an unsorted set of possible next activities by DeciClareEngine. Hence, it does not always offer much direct guidance to the process actors. The needed guidance can be added by way of a recommendation generator that uses past process instances to rank the available next activities. Section 5.2 provides an implementation of forty strategies to generate next-activity recommendations and describes the evaluation and comparison of these strategies. A recommendation generator based on these implementations is arguably not a part of the BPE-component itself, but rather of the Process Analytics-component of a BPMS (see section 1.1) as an additional layer on top of the BPE. The combination of these two subcomponents provides full operational support to process actors during the execution of a process instance. The BPE ensures that the execution conforms with the explicit knowledge of the given process and decision model, while the recommendation engine offers insight and guidance into the implicit experiences that are contained by past executions of the process.

5.1. A GENERIC FRAMEWORK FOR FLEXIBLE AND DATA-AWARE BUSINESS PROCESS ENGINES

5.1.1. Introduction

Loosely framed processes are processes that can be executed in a large, but finite, number of ways with all possible activities known in advance (van der Aalst 2013; Di Ciccio, Marrella, et al. 2015; Mertens et al. 2017). Example domains are healthcare (e.g., diagnosis and treatment of a patient), legal (e.g., preparing, arguing and pleading a case), helpdesk (e.g., finding a satisfactory solution) and air traffic control (e.g., deciding the priority for the arriving/departing planes). To model loosely framed processes, declarative process modeling languages are the preferred choice due to their implicit support for process flexibility (van der Aalst 2013; van der Aalst et al. 2009; Goedertier et al. 2015; Mertens et al. 2017). At the same time, these processes can also be classified as knowledge-intensive (KiP) (Di Ciccio, Marrella, et al. 2015; Mertens et al. 2017): they require knowledge workers (e.g., doctors) to leverage their domain knowledge in search for an appropriate process variation for each case. Therefore, declarative modeling language should also incorporate some form of decision logic, as this allows the domain knowledge to be captured that is essential to the execution of these processes. Declarative process models are often difficult for humans to create and to understand (Haisjackl et al. 2014). The innate complexity of the processes themselves is the leading cause, however, the absence of a clear path or flow through the models is a strong second. It is easy to lose sight of the overall process when confronted with a large set of constraints, of which each only applies to specific contexts that can vary widely from one constraint to the next. Moreover, it is hard to capture the tacit knowledge of knowledge workers, accumulated during years of experience, in an explicit process model manually (e.g., through interviews). Process mining techniques can be used to overcome this issue by (semi-)automatically creating models from historic data logs of the process, if available. With a model of the process available, whether it be an original or redesigned version of a manually or automatically created model, the next step is to make sure

that future process executions are in conformance with the model. However, due to the aforementioned understandability issues, doing this manually is challenging.

A Business Process Engine (BPE) enables process actors to correctly execute process models by enacting them (Dumas et al. 2005). A BPE manages the workflow by monitoring and validating the execution, by showing an overview of the partially executed process instance and the restriction that currently apply based on the model, and by notifying the responsible process actors or automated services when needed. The complexity of the models is less of an issue for BPEs, because computers are well equipped for this type of complexity. Although there exist many BPEs for imperative process models, the offerings for declarative models are much more limited. Even more so if the BPE needs to be able to handle decision logic captured in the models, which represents the knowledge that is so vital to the correct execution of loosely framed KiPs in real-life.

This paper is part of a design science project and, more specifically, a design science cycle to develop the architecture of a business process management system for loosely framed KiPs. In previous work, we have created a modeling language specifically for this type of processes, called DeciClare (see chapter 2), and developed an automated framework for creating DeciClare models by way of process mining (see chapters 3 and 4). The contribution of this paper is a generic and high-level framework to support the execution of declarative process models with a data perspective. This is achieved by automatically generating sets of activities that are allowed, conditionally allowed and disallowed as the next activity at each point during the execution of the process. The framework is generic because it does not assume a specific modeling language, but rather a large family of languages that can satisfy the language requirements stated in this paper. Consequently, the actual constraint validation techniques and corresponding computational complexity are out of scope as they are language depend. The goal is to pave the way for flexible decision- and process-aware support for loosely frame and KiPs.

The remainder of this paper is structured as follows. Section 5.1.2 presents the research methodology used in this paper. The problem statement and the requirements for the solution artifact

are presented in section 5.1.3. In section 5.1.4, the generic framework is proposed based on the requirements from section 5.1.3. Subsequently, the DeciClare language is used to demonstrate a proof-of concept implementation of the framework in the context of a realistic healthcare process in section 5.1.5. Section 5.1.6 gives an overview of the related research on this topic. In section 5.1.7 the paper is concluded and directions for future research are presented.

5.1.2. Research Methodology

This paper applies the Design Science Research Methodology (Hevner et al. 2004; Peffers et al. 2007) to develop a high-level framework to support the execution of declarative process models with a data perspective (i.e., the design artifact). The problem and requirements for the design artifact are defined in the next section. Because there is not a lot of literature on BPEs, the requirements are based on scattered information from literature, practical examples of BPEs from existing BPMSs and common-sense extensions of these principles.

The design artifact will be demonstrated by way of a prototype implementation. This should prove that a fully functional implementation of the framework is practically achievable and give an idea of how it would work.

5.1.3. Problem Statement and Solution Requirements

Declarative process models consist of many individual constraints and each of these constraints possibly applies to a different context. The result is that it can be very difficult for a user to quickly get an overview and find the process variation suitable for a certain process instance. This is largely due to the inherent nature of the processes being modeled, namely loosely framed KiPs. For example, consider the sheer number of possible diagnoses and treatments to the disposal of an emergency services doctor when a new patient arrives. The decision to follow a specific process variation is taken incrementally by the doctor during the process execution by choosing one or maybe a couple of activities to do next at a time. Because declarative models are hard to read or understand, there is no straightforward way in which they can be enacted directly by users.

A BPE is a software tool that offers supports the execution of a business process model. In this case, it should enact a declarative process model by creating a simplified summary for the users based on the specific process instance context. This summary will need to satisfy the following engine requirements (ER):

ER1. The workflow control data containing the current state of each process instance (i.e., the already executed activities and used resources) should be available

ER2. The workflow relevant data of each process instance (e.g., age and gender of patient, presented symptoms...) should be available

ER3. All behavior allowed by the declarative model should be allowed

ER4. All behavior that violates the declarative model should be prohibited

The four engine requirements above represent the foundation of a BPE; the typical responsibilities of a BPE in a BPMS in literature (Leymann and Roller 2000; Montali 2010; Weske 2012; Wohed et al. 2009) and the available real-life examples (e.g., jBPM¹¹, Activiti¹², DCR Graphs simulation engine¹³, etc.). Depending on the expressive power of the declarative modeling language used to create the model, the resulting BPE can incorporate the control-flow, data and resource perspectives of the process. For loosely framed KiPs it remains essential that the freedom allowed by the given declarative model is not reduced by the BPE. The model does not need to be complete and the available constraints should be followed, but it should always present every option available according to the model. The BPE is not an expert system that replaces knowledge workers, but rather a tool to make their job easier.

To make the BPE more user-friendly, it would be useful to provide the user with an explanation of why certain activities are prohibited or will impose future restrictions and with some form of task list or description containing the constraints that still need to be satisfied by executing future

¹¹ <https://www.jbpm.org/>

¹² www.activiti.org - Open Source Business Automation

¹³ <https://dcrsolutions.net/>

activities. The following engine requirements are based on common sense extensions of the typical responsibilities of a BPE as described in literature (Bernardi et al. 2012; Haarmann et al. 2015; Lanz, Posenato, et al. 2014; Pryss et al. 2015) and features real-life BPEs to improve the user-friendliness (e.g., the reminder functionality of Activiti):

ER5. The reasons why certain activities are not possible at certain times or in certain contexts should be available in the form of the constraint of the declarative model that would be violated

ER6. The restrictions that need to be dealt with later in the execution of the instance should be available to the users (e.g., execute activity X within 5 hours)

ER7. A warning should be given for all behavior that potentially could lead to a live- or deadlock (prevention is even better, but not always computationally feasible depending on the used declarative process modeling language)

BPEs require certain information about the process and its instances to be able to deliver useful results. Early BPEs were only concerned with the activities in a process and the relations between them (van der Aalst 1998). Newer BPEs also add information about the resources involved in the process (e.g., personnel). ER1 and ER2 already require the BPE to keep track of the state and specific context of each process instance, but additional requirements are needed on the process model level to be able to create a data-aware declarative BPE. We summarized the informational needs in 8 language requirements (LR). If the process modeling language used to create the process model does not support these LRs, it might be possible to provide this information separate from the model.

LR1. A set of all activities of the process

LR2. A set of all resources involved in the process

LR3. A set of all data elements involved in the process

- LR4. The capabilities of each resource (e.g., a dialysis machine can be simultaneously used in maximally two process instances)*
- LR5. The availability of each resource (i.e., how many of this type of resource and when they are available for a process instance)*
- LR6. The relationships that apply between resources (e.g., a nurse can take on the role of both a scrub nurse and an OR nurse)*
- LR7. The relationships that apply between the activities and/or the resources (e.g., the patient must be registered before doing an examination and doing an examination requires a doctor and an examination room)*
- LR8. The decision logic, based on the data elements from LR3, that governs the relationships from LR5 and LR6*

5.1.4. Data-Aware Declarative Process Enactment Framework

The ER1 and ER2 are related to the process instance and are stored on the enactment level. ER1 can be satisfied by keeping a history for each process instance of steps performed and resources utilized. ER1 requires access to the information system of the organization to get the necessary data values and real-time resource usage stats. This is not trivial in real-life, but for the purpose of this paper we will assume that this data is directly available in the same format as defined in the declarative model.

The satisfactory execution of a process instance requires the joined efforts of the BPE and the knowledge workers of the process, because declarative models do not define explicit pathways to follow. The BPE takes on the role of relevant information provider and the knowledge workers use this information to make the decision on what to do next. The framework at the core of the BPE consists of two phases. In the first phase, the BPE uses the workflow relevant and control data (ER1 and ER2) to calculate the parts of the declarative model that apply specifically to the executing instance by looking at the decision logic and preconditions for each constraint (see Table 21). Note that vacuously satisfied constraints (Maggi et al. 2016) should be included (perhaps separately) as

they convey important information on possible future restrictions. In the second phase, the BPE calculates which activities are available based on this relevant subset of the process model (see Table 22). It is then up to the knowledge workers to make the actual decision on how to proceed. The decision to execute a certain activity next will prompt an update of the workflow control data and likely generate additional data, which in turn triggers the BPE to redo the two phases. This is repeated until an ending is reached.

The second phase creates five (unsorted) sets of activities, which we will call tiers:

- (A) A set consisting of the activities that are available for execution as the next activity without any future restrictions applying.
- (B) A set consisting of the activities that are available for execution as the next activity, but with future restrictions applying. These future restrictions come in the form of activities that are required or prohibited to be executed in the future so that the partial instance can be finished correctly.
- (C) A set consisting of the activities that are available for execution as the next activity (i.e., tier A or B), but only if the active user is changed to another user.
- (D) A set of the activities that are not available now but will be after a specified amount of time has elapsed (and will subsequently move to tier A or B).
- (E) A set of the activities that are not available for execution as the next activity.

<i>Input:</i> a declarative process model, the current state and current context of a process instance		
#	Description	Req.
1.	Make a set of all constraints in the declarative model	LRLR 1-LR8
2.	Check which constraints are active for the process instance by comparing the process instance data with the decision logic of the constraints. Remove the inactive constraints from the list.	ERER 1
3.	Remove all constraints for which the prerequisites are not satisfied by the history of executed activities and used resources of this process instance.	ERER 1
<i>Output:</i> a set of all constraints applicable to the given process instance		

Table 21. The first phase of the framework of a declarative business process engine

<i>Input:</i> a declarative process model and the output of the first phase	
1.	Make set of all activities defined in the model, called the potential next-executed activities.
2.	For each activity in the set of potential next-executed activities: For each constraint from the set of applicable constraints: Validate that executing this potential next-executed activity would not violate the constraint. If only satisfied if executed after waiting a finite amount of time, store the constraint in the set of time-violated constraints for this potential next-executed activity.
	Else if only satisfied if future requirements are fulfilled, store the constraint in the set of activity-restricting constraints for this potential next-executed activity
	Else if only satisfied if the active resource (=current user of the system) is a different resource than the current active resource, store the constraint in the set of resource-restricting constraints for this potential next-executed activity.
	Else store constraint in the set of violated constraints for this potential next-executed activity.
3.	For each activity in the set of potential next-executed activities: If the violated constraints set is not empty, add activity to tier E.
	Else if the time-violated constraints set is not empty, add activity to tier D.
	Else if the resource-restricting constraints set is not empty, add activity to tier C.
	Else if the activity-restricting constraints set is not empty, add activity to tier B.
	Else add activity to tier A.
<i>Output:</i> tiers A, B, C, D and E	

Table 22. The second phase of the framework of a declarative business process engine

These five tiers give an overview of the allowed and prohibited behavior. ER3 and ER4 are satisfied by allowing only the activities in tiers A and B to be executed next. All activities in tier B are subject to restrictions that need to be dealt with later in the process. In contrast to the activities in tier A, the BPE cannot guarantee that the activities in tier B will not lead to a live- or deadlock. By placing them in a separate tier, this serves as a warning for the user to keep an eye on these restrictions (ER7).

ER5 calls for information to help the knowledge workers executing the process to understand why an activity is in a specific tier. To fulfill this requirement, each activity in tiers C, D and E will be accompanied with the constraints that would be violated by executing the activity next, which is a byproduct of the second step of the second phase. ER6 relates to the possible restrictions that apply to a certain partial instance and will need to be satisfied before the instance can finish in a correct way (e.g., X needs to be executed because Y has been executed). Fulfilling this requirement is a twofold process. Firstly, the constraints that result in these restrictions must be gathered. This set is already being composed as part of the second step of the second phase: the set of activity-restricting constraints. Secondly, the information needs to be made available to the user. The user needs to be

able to view this set of restrictions, both for the current partial trace as well as for each activity from tier B (= the restrictions that would apply if this activity is selected as the next executed activity).

Additionally, some extra information about resource restrictions, time delays and deadlines that apply during the process execution can be added. Each activity in tiers C and D can be annotated with, respectively, the required user and the time delay that needs to be respected before it can move to tier A or B. The delay for a certain activity can be calculated by comparing the remainder of the current time minus the activation time, and the delays as specified by the constraints in the set of time-violated constraints for that activity from the second phase. Activities in tiers A and B can be annotated with the deadlines, if applicable, before which each activity needs to be executed. The relevant deadline can be calculated as the lowest deadline from the constraints in the set of activity-restricting constraints from the second step of the second phase, compared to the current time and taking into account the activation time of each of these constraints. Also, the subset of constraints from the full declarative model that apply to the current partial trace and for each activity in tier A and B can be made available, so that the user can review only those constraints that actually matter in this process instance. Again, we would preferably include the vacuously satisfied constraints separately, as these could be interesting when considering future restrictions.

Finally, there is one more possible scenario to be considered. As constraints can have deadlines before which they need to be satisfied, it is possible that a deadline is not satisfied (i.e., the necessary activities are not executed in time). This causes a deadlock. A BPE cannot directly prevent this, as user input is required for executing activities and it is the user's responsibility to monitor the deadlines as shown by the BPE.

5.1.5. Demonstration

As a proof-of-concept, we will use the DeciClare language to satisfy the language requirements from section 5.1.3 and demonstrate a working implementation of the data-aware declarative process enactment framework proposed in the previous section.

5.1.5.1. Process Definition: DeciClare

DeciClare is a mixed-perspective declarative process and decision modeling language that specifically targets loosely framed KiPs (see chapter 2). It can be used to create detailed models of the control-flow, data and resource aspects of these processes. This paper uses a direct instantiation (i.e., a temporary visual syntax in a textual format) of the abstract syntax, as a real visual syntax has not yet been developed at this time. The foundation of DeciClare is the Declare language and most of its extensions (Jiménez-Ramírez and Barba 2013; Pesic 2008; Westergaard and Maggi 2012), which are either incorporated in a direct or generalized way. Therefore, Declare models can be easily converted to equivalent DeciClare models. Additionally, more perspectives were added as well as concepts from the DMN language (Object Management Group 2014). The latter makes the resulting language ‘data-aware’, because data values can be used to (de)activate constraints based on the specific execution context, and capable of modeling the decision logic that typically governs KiPs. DeciClare supports 36 constraint templates (i.e., types of constraints) spanning four template classes: existence, relation, resource and data. 26 of these templates have time parameters expressing the time interval in which the constraint applies.

The current implementation of the DeciClareEngine-tool presented in this paper supports the 18 most important constraint templates of DeciClare (see Appendix F). A DeciClare model can contain all the information necessary to satisfy LRLR1-LR8 from section 5.1.3. Activities, resources and all their relations can be fully modeled, and the constraints can be enriched with activation and deactivation decisions, based on the data elements, to allow for a level of detail that closely matches reality.

5.1.5.2. DeciClareEngine

The DeciClareEngine tool (github.com/stevmert/DeciClareBPMS/blob/master/engine/releases/DeciClareEngine.jar, the code can also be found on this GitHub) has been developed to demonstrate how the generic framework from section 5.1.4 can be implemented. It is a data-aware declarative

BPE that takes a DeciClare model as input. This means that it supports the control-flow, decision logic, time restrictions, resource authorizations and resource usages of the given process model.

When starting the BPE, the user first needs to select the role he or she will take on during the process instance before the main interface opens. At the top of the screen (left side of Fig. 30), the current time and user (= current resource) can be found as well as the current partial trace. The tiers A to E are displayed in the middle of the screen. And at the bottom of the screen, all data elements used in the model are listed with the option to add or change them for the current instance. The user can execute an activity from tier A or B by clicking on it. This prompts a screen where the user can enter the duration of the activity and change the data elements as part of the activity execution (right side of Fig. 30). The ‘Explanation’-button next to the activities in tiers B, C, D and E presents the user with a set of the specific constraints that caused the activity to be placed in that specific tier (i.e., future restrictions or violations). Note that delays and deadlines are signaled next to the activity name and that their current status can be checked at any time as part of the explanation. To get the subset of constraints that are currently active, the user can click on the ‘Relevant model’-button at

Fig. 30. The main interface (left), with added explanations, and activity execution dialog (right)

the bottom of the screen or next to the corresponding activity. Vacuously satisfied constraints are included in this summary, although they are not yet displayed separately.

As the example model, we will use the healthcare process for treating arm-related fractures introduced in section 2.5.1 (also see Appendix D). It is a realistic, but simplified, representation of the registration, diagnosis and treatment of patients with one or more (possible) fractures of the fingers, hands, wrists, forearms, upper arms, shoulders and collarbones in an emergency department of a hospital. The DeciClare model contains 229 constraints, of which 183 are supported by the current version of the tool. The process instance used to demonstrate the tool is of a 24-year-old woman who enters the emergency services with pain in her wrist after a bicycle accident. Every activity of the process is listed in the first column of Table 23. The other columns are snapshots (Sx) (i.e., 0-13) showing in which tier each activity was placed by the BPE at that time in the process.

	Receptionist			Doctor										Receptionist	
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	
Register patient	B	B	E	E	E	E	E	E	E	E	E	E	E	E	
Take medical history of patient	C	E	C	B	B	B	B	B	B	B	B	B	C	E	
Clinical examination of patient	C	E	E	E	B	B	B	B	B	B	B	B	C	E	
Doctor consultation	E	E	E	E	E	B	B	B	B	B	B	B	C	E	
Take X-ray	E	E	E	E	E	B	B	B	B	B	B	B	C	E	
Take CT scan	E	E	E	E	E	B	B	B	B	B	B	B	C	E	
Prescribe NSAID painkillers	E	E	E	E	E	B	B	B	B	B	B	B	C	E	
Prescribe SAID painkillers	E	E	E	E	E	B	B	B	B	B	B	B	C	E	
Prescribe anticoagulants	E	E	E	E	E	B	D	D	D	D	D	D	D	E	
Prescribe stomach protecting drug	E	E	E	E	E	B	B	B	B	B	B	B	C	E	
Prescribe rest (=no treatment)	E	E	E	E	E	B	B	B	B	B	B	B	C	E	
Apply ice	C	E	C	B	B	B	B	B	B	B	B	B	C	E	
Apply cast	E	E	E	E	E	E	E	E	B	B	B	B	C	E	
Remove cast	E	E	E	E	E	E	E	E	E	E	E	E	E	E	
Apply splint	C	E	C	B	B	B	B	B	B	B	B	B	C	E	
Apply sling	C	E	C	B	B	B	B	B	B	B	B	B	C	E	
Apply fixation	E	E	E	E	E	E	E	E	B	B	B	B	C	E	
Remove fixation	E	E	E	E	E	E	E	E	E	E	E	E	E	E	
Apply bandage	C	E	C	B	B	B	B	B	B	B	B	B	C	E	
Apply figure of eight bandage	E	E	E	E	E	E	E	E	B	B	B	B	C	E	
Perform surgery	E	E	E	E	E	E	E	E	B	B	B	B	C	E	
Apply intra-compartmental pressure monitor	E	E	E	E	E	B	B	B	B	B	B	B	C	E	
Let patient rest	C	E	C	B	B	B	B	B	B	B	B	B	B	E	
Stay in patient room	E	E	E	E	E	B	B	B	B	B	B	B	C	E	
Unregister patient	E	E	B	C	C	C	C	C	C	C	C	C	A	E	

Table 23. An overview of the activities and the tier in which they were placed for each snapshot

S0 is the moment that the patient walks in the emergency room, with the receptionist as active user. The receptionist is not authorized to execute 7 of the 8 available activities, all of which come with future restrictions. The triage nurse takes a quick peek at the patient and concludes that her condition is not life-threatening, triggering the corresponding data event. This reduces the number of available activities to just one (S1), because only in the worst cases it is possible to skip the registration at the start. Next, the patient is registered at the reception, during which the receptionist sets the four data attributes relevant to the process (i.e., been to this hospital before, older than 14 years old, not bedridden and conscious) and the used resources (i.e., receptionist and reception desk). The user is currently asked to input the duration, data events and resource events of the executed activity (right side of Table 23), but in real settings this data should be extracted directly from the applications used to capture this data. Once registered, 7 activities become available, but the receptionist is only authorized to unregister the patient (S2). At this point, the active user is switched to a doctor, who is authorized to execute the 6 other activities (S3). By looking at the future restrictions that apply to the current partial instance, the doctor can see that several activities are now mandatory for this instance: unregistering the patient, taking a medical history, doing a clinical examination, taking an X-ray, applying ice, choosing a treatment and giving some pain relief. The doctor decides to proceed by letting a nurse take the patient's medical history, triggering two more data events (i.e., no current cast or fixation on the wrist in question) and one resource event (i.e., nurse used). Taking the medical history of the patient unlocks an additional activity in tier B: doing a clinical examination (S4). This is also the activity the doctor chooses to do next, which in turn triggers four more data events (i.e., suspected fracture of wrist, swollen, in pain and not an open fracture) and two resource events (i.e., uses doctor and exam room). S5 has 18 available activities. Because the doctor indicated that the patient is in pain, she can now receive some pain relief. After giving her a NSAID painkiller, 17 activities become available (S6). Note that prescribing anticoagulants is in tier D because there must be at least 12 hours between receiving NSAID painkillers and receiving anticoagulants. 12 hours later, this activity would move back to tier B.

By taking another look at the future restriction that currently apply, the doctor can see that three activities are still mandatory for this instance: unregistering the patient, applying ice and taking an X-ray. The former is straightforward, but the latter two are due to the specific context of the case. Ice treatment is needed because the wrist is swollen and the X-ray to take a closer look at the patient's wrist. The doctor decides to do the ice treatment first. S7 offers the same options as the previous one and now the X-ray is selected, triggering two more resource events (i.e., nurse and X-ray room used). This results in four additional activities becoming available (S8). A doctor consultation to discuss the scan is now a mandatory, as stated in the current restrictions, so the doctor does this consultation next. During the consultation (using a doctor and an exam room), the doctor confirms the suspected wrist fracture by looking at the X-ray photo but, luckily, also that it is not a displaced or unstable fracture. This means that surgery is not required. The same activities are available in S9, yet it is now mandatory to apply a sling and a cast/splint in the future. The doctor decides to apply a splint next, which triggers another resource event (nurse used). Again, the same activities are available afterwards (S10). So now the doctor applies a sling to support the splinted wrist of the patient. In S11, the same activities are available, but the only current restriction now is that the patient will need to be unregistered in the future. The doctor decides that the patient can go home, so she is asked to go and unregister at the reception. The active user is now switched to the receptionist again (S12), which moves unregistering the patient from tier D to tier A (and 19 others from B to D). This means that no restrictions will remain after its execution. Finally, the receptionist unregisters the patient, triggering two more resource events (receptionist and reception desk used). No activities are available in S12, and the instance is in a conforming state according to the process model, so the user gets a message that the process has ended successfully.

5.1.6. Related Work

The existing BPEs can be classified according to the predictability-spectrum of the processes for which they are best suited. In this section, the differences between the DeciClareEngine-tool, used

in the demonstration of this paper, and each of the existing BPEs that target loosely framed processes will be highlighted in more detail.

The DECLARE framework is based on the Declare language (Pesic 2008), for which efficient techniques for the actual validation of (MP-)Declare constraints have been proposed (Burattin et al. 2016). Everything that can be expressed in Declare, can be expressed in the DeciClare language used in this paper (Mertens et al. 2017). Yet, the opposite is not true as DeciClare adds a data and resource perspective as well as an explicit concept of time in the form of deadlines and delays. So, compared to the DECLARE framework, DeciClareEngine additionally takes decision logic, resources, deadlines and delays into account. More recently, a BPE for the MP-Declare language was proposed that maps the constraints to Alloy and uses the corresponding satisfiability solver (Ackermann et al. 2018). This approach can prevent live- and deadlocks. However, it is only feasible for very small models (e.g., 10 constraints) due to the computation complexity that results from the state space explosion. DeciClareEngine has no support for this yet, but this feature will be revisited in the future. Models of loosely framed KiPs are typically big and complex. Therefore, we will focus on investigating partial or heuristic versions of these features in future work.

ReFlex (De Carvalho et al. 2013) is a BPE that uses a graph-based rule engine to overcome certain limitations of Linear Temporal Logic (LTL), as is used by the Declare language. It uses 7 types of edges that correspond to 6 constraint templates of Declare and 1 extra type of constraint. DeciClare, and therefore DeciClareEngine, also supports these 7 types of constraints and additionally offers support for time delays and deadlines, and a data and resource perspective. The selling point of ReFlex is also the detection of live- and deadlocks.

DCR Graphs have an online platform¹⁴ that includes a BPE as part of the graph editor (for simulation). The DCR language has 6 relations (roughly equivalent to the Response and Precedence templates of DeciClare) and a clear visual syntax with support for hierarchical modeling. The

¹⁴ <https://dcrsolutions.net/>

simplicity of DCR Graphs is good for human understanding but limits its expressive power, although a data perspective (Mukkamala 2012) and some time-concepts (in DCR 2.0) have been added recently. DeciClare has additional support for more complex constraints, general context data elements, changing values of data attributes during the process execution and deadlines. On a functional level, the DCR Graphs BPE is quite similar to the prototype presented as demonstration of this paper, albeit based on a less expressive language but with a much more polished interface. There is no academic description of how the DCR Graphs BPE works internally, although a demonstration is provided in Marquard et al. (2015). The contribution of our paper is the generic framework to base this kind of BPE on, which has not been published before.

5.1.7. Conclusion and Future Research

This paper proposes a generic framework for creating data-aware declarative BPEs and demonstrates it by applying it to create a BPE for the DeciClare process modeling language. A BPE based on the proposed framework can support the execution of a real-life loosely framed KiPs, taking into account the control-flow, data and resource perspectives and corresponding decision logic governing them.

Due to the complexity and understandability issues of declarative languages, declarative models are seldom useful during the execution of processes. A BPE can be used to support the process actors during the execution of the process model by offering a process-aware view on what has been done already, what can or cannot be done next and what will need to or cannot happen in the future. The main advantage of using a BPE for the execution of a process is, of course, the certainty of conformance to a predefined process model. However, it can also be the foundation for a truly flexible information system, because it enables the rapid deployment of process changes in dynamic environments. Just make the necessary changes to the underlying process model. The BPE allows for these changes to be communicated to all process actors and at the same time enforce that the changes are being implemented correctly.

In addition to offering runtime support, BPEs are also an essential tool for process simulation. Simulation makes it possible to analyze the impact of different runtime decisions or changes to the process model itself on the process outcome or characteristic. Another way to use model simulation, would be to validate a model that is the result of process mining or interviews. Historical or fictitious process instances can be simulated using the BPE, allowing domain experts to validate the process model in specific and more relatable contexts. This type of model validation is similar to how Benevides et al. (2010) uses Alloy to simulate OntoUML models to validate that the model allows all wanted interpretations, while excluding the unwanted ones. It is also a powerful tool for educational purposes. The current education for doctors and other medical personnel leans heavily on internships so that the trainee can learn on the job from more experienced personnel. An educational simulator offers the trainee an additional tool to learn how things are done in a certain hospital and gain valuable experience in a risk-free environment. By using a mined process model that includes decision logic, it even has the potential to reveal (some of) the tacit knowledge of experienced colleagues to the trainee without a need for any direct interaction with those colleagues or patients.

The current design cycle of our research project is not finished yet. DeciClareEngine will be further refined by: 1) Adding support for the missing constraint templates and deactivation decisions, 2) Adding support for dynamic resource availabilities on process instance level, 3) Adding support for parallel execution of activities and 4) Investigating partial live- or deadlock detection or prevention mechanisms. A thorough comparison of the advantages and disadvantages of process execution with and without the use of DeciClareEngine is also planned for future research.

5.2. COMPARING STRATEGIES TO GENERATE EXPERIENCE-BASED CLINICAL PROCESS RECOMMENDATIONS THAT LEVERAGE SIMILARITY TO HISTORIC DATA

5.2.1. Introduction

Healthcare processes can be subdivided into two groups: medical diagnosis/treatment processes and organizational/administrative processes (Lenz and Reichert 2007). In this paper we focus solely on the former. These kinds of processes are characterized as dynamic, multi-disciplinary, flexible, human-centric and knowledge-intensive processes (Mertens et al. 2017; Rebuge and Ferreira 2012). Consequently, the process actors executing the process (i.e., doctors and other medical personnel) are constantly confronted with decisions on what to do next from a large and diverse set of options. Decision making can be very demanding and even overwhelming, certainly for less experienced process actors. The current IT infrastructure remains lacking in support for decision making (Meier et al. 2015). This is one of the reasons for requiring internships for new personnel. It gives them time to get a feel of how things are done in a certain department by following around more experienced personnel. However, these intern periods are time and effort consuming, not to mention expensive. Adding an additional channel for externalizing the knowledge of more experienced process actors as guidance, without them having to do anything extra, would have clear benefits (Mundbrod and Reichert 2017). Not only for less experienced process actors, but also for the other process actors. The result will be that the offered service will be more uniform, intern periods can be shortened, and process actors get relieved of some of the daily pressure.

In this paper, we take the first step towards offering such guidance in the form of recommendations for flexible knowledge-intensive processes, and more specifically healthcare processes concerning the medical diagnosis and treatment of patients. The purpose of these recommendations is not to steer the process according to some optimization criteria, but rather to offer a direct reflection of the experience encapsulated in previous executions of the process. The results are returned to the users as process recommendations: a ranked list of the activities that are

predicted to potentially be executed as the next activity and the corresponding probability estimations based on the most similar previously completed executions. The performance of a variety of strategies was measured and compared. The strategies can differ in what information they focus on, but they all leverage similarities between the currently executing process instance and the available historic data. The research questions (RQ) of this paper are the following:

- RQ1. Which prediction strategies perform well for knowledge-intensive and flexible healthcare processes?
- RQ2. What kind of precision and consistency can be achieved and in what amount of computation time?
- RQ3. How much data is needed to for near-optimal precision?
- RQ4. Which strategies could be useful benchmarks to evaluate more advanced techniques (e.g., neural networks) in the future?

The strategies were implemented and applied to three different healthcare data sets: two real-life event logs and one artificial log. They are evaluated on the accuracy and consistency of the recommendation rankings (i.e., average accuracy, multi-class brier score, log loss and rank score) and the required computation time.

The remainder of this paper is structured as follows. Section 5.2.2 gives an overview of the related research. The relevant terminology and formal problem definition are presented in section 5.2.3. In section 5.2.4, the applied research methodology is presented. The prediction strategies used to generate recommendations are explained in section 5.2.5. Section 5.2.6 describes the setup and results of experiments to evaluate the prediction strategies. The results of the experiments are discussed in section 5.2.7. And finally, the paper is concluded and directions for future research are suggested in section 5.2.8.

5.2.2. Related Work

The domain of process recommendations as a way to offer guidance to the process actors is still in its infancy. The prediction, ranking and probability estimation of potential next activities as a direct reflection of the experience encapsulated in previous executions of the process is, to the best of our knowledge, yet to be tackled. However, process recommendations have been proposed with other goals in mind. Schonenberg and Weber (2008) use three different trace abstractions to generate recommendations based on historic executions that minimize the cycle time of a partially executed process instance. The trace abstractions (prefix, set and multi-set) have also been used in this paper. Barba et al. (2013) propose a recommendation system for process-aware information systems that generate an enactment plan for the remainder of a partially executed process instance using constraint-based approach with a declarative process model as input. The differences with this paper are that only one enactment plan is proposed at any one time, so no more than one recommended next activity per available resource (i.e., no ranking), and an enactment plan specifies the remainder of the process instance (i.e., all recommended remaining activities in a specified order). The enactment plans are also optimized according the performance goals of the process, whereas the goal of this paper is to reflect the current way of working.

The related domain of predictive process monitoring is concerned with the continuous generation of predictions (often referred to as next-element predictions) of what activities should be performed and what input data values to provide, so that the likelihood of violation of business constraints is minimized (Maggi, Di Francescomarino, et al. 2014; Metzger et al. 2015). A business constraint can be a business goal, a desired outcome or a range for a key performance indicator (KPI) (e.g., instance duration should be less than 24 hours). The prediction of the next, the last or all remaining activities as well as the similarity to previously executed instances is used to estimate the likelihood of compliance based on the current state of the instance and the available historic data. The key difference to this paper is that the goal of predictive process monitoring is typically to indicate when special attention, or even an intervention, is needed to prevent the process instances from reaching

an undesirable state. So, the prediction of the likelihood of compliance to some predicated constraint is of primary concern, more so than the precision of the next-activity prediction. Therefore, the prediction of the next activity is typically just a means to an end and not necessarily conveyed to the process actors. In this paper, our focus is solely on the prediction, ranking and probability estimation of potential next activities as a direct reflection of the experience encapsulated in previous executions of the process and we make no attempt to optimize the likelihood of achieving some business goal or to keep some KPI within an optimal range. In real-life applications, the ranking and probability estimations of the potential next activities proposed in this paper would be made available to process actors as a form of guidance. As a result, the primary focus is on the precision of the predictions as well as the relative ranking and the robustness of the ranking. Some predictive process monitoring systems can also provide recommendations as a way of intervening when potential goal violations loom. This is an attempt to sway the process actor to follow historic executions that do satisfy the predicated constraint, as opposed to other historic executions that do not. The difference here is that they value certain similar historic executions more than others based on their outcomes, as opposed to just the direct similarity between the current execution and historic executions as is the goal of this paper.

Several techniques used in predictive process monitoring could be useful in the context of this paper, but it is difficult to compare the results from those papers with ours as the evaluation setup and criteria are different and the implementation, if available, often needs some adjustments to align it with the goals of this paper. Di Francescomarino et al. (2016) apply a clustering technique and a classifier to estimate whether a given predicate will be satisfied upon completion of an executing process instance. Evermann et al. (2016) use recurrent neural networks with long short-term memory to predict the remaining time to completion and the next activity of partially executed process instances and Tax et al. (2017) further investigated its applicability. van der Spoel et al. (2013) use process mining techniques to predict the next activity and a classifier to predict future cost and duration of a partially executed process instance.

Another related domain is that of trace clustering. Trace clustering techniques are often applied to create clusters of similar completed process executions before process mining techniques are applied to create a model of each cluster. These techniques should not be applied to cluster a partially executed instance in clusters of completed executions, as this is a completely different problem. However, some techniques use traces abstractions as a first step towards comparing the similarity of traces and we have implemented several of these as strategies in this paper. From Song et al. (2013) we used the activity and transition trace profiles, which make abstraction of a trace by keeping only a set (or vector/matrix) of activities executed or activity transitions used in the trace. From De Leoni et al. (2016) we used the control-flow manipulations CFP1 (i.e., the number of executions of a certain activity up to that point in the trace) and CFP3 (i.e., the activity transitions up to that point in the trace) as abstractions of a trace. Additionally, many cluster distance measures can be also be used as similarity metrics. We started from the distance measures in Song et al. (2009) (e.g., Euclidean, Jaccard...) and added several others from different sources. The resulting strategies in a way make just one cluster with the abstraction of the partially executed instance as center. To calculate cluster distance (i.e., the similarity) between a partial and a completed instance, we introduced the concept of a variable-position similarity scorer.

5.2.3. Terminology and Formal Problem Definition

First, some definitions of terms or abbreviations that will be used throughout the paper:

- *Activity*: an identifiable action performed by a process actor as part of achieving a goal of the process. It can consume both time and resources and can produce data.
- *Activity event*: a timestamp that delineates a specific milestone (e.g., start or end) of the execution of an activity. The event itself does not consume any time or resources. IT-systems typically store these events, making it possible to reconstruct the chronology and durations of the actual activities of the process.

- *Data event*: a timestamp at which a specific data value was produced during the execution of a process. A data event can also arise from external sources, so it does not necessarily have to be produced during an activity. IT-systems typically store these events, making it possible to reconstruct the chronology of the actual data produced during the execution of the process.
- *Trace*: a combination of chronologically ordered activities and data events that correspond to one execution of the processes (e.g., the diagnosis and treatment of a patient).
- *Prediction points in a trace*: a trace is divided into prediction points, each coming right after the completion of an activity and one just before the start of the first activity in the trace. For example, a trace $\langle A, B \rangle$ has three prediction points: before A, after A and after B. For simplicity, we did not consider each moment in time between the execution of activities as a separate prediction point.
- *Position in a trace*: each prediction point in a trace has a position. Consequently, the prediction point before the execution of the first activity is position 0, after the execution of the first activity is position 1, and so on.
- *Multiset*: a set in which elements are not necessarily unique (i.e., a list without a specified order).
- *NextActs*: the set of potential next activities for every prediction point. Let $NextActs = [A_1, \dots, A_a, A_{a+1}]$ with a the number of executable activities and A_{a+1} representing the end of the trace (i.e., no more activities executed).
- *HT* (=historic trace): a trace of a previously completed process instance. The set of historic traces is used by the scorers to calculate recommendations for new partial traces. This set is also referred to as the learning set. Let $H = [HT_1, \dots, HT_k]$ with k the number of traces in the learning set and $HT_i = [A_{HT_i(1)}, \dots, A_{HT_i(m)}, A_{a+1}]$ with m the number of executed activities in trace HT_i .

- *HPT* (=historic partial trace): a partial version of a *HT* where the activity and data events after a certain position in the *HT* are not considered.
- *CPT* (=current partial trace): the partial trace for which the recommendations are being calculated. Let $CPT = [A_{CPT(1)}, \dots, A_{CPT(n)}]$ with n the number of already performed activities.

Now then we can provide a formal problem definition:

For each prediction point, predict $A_{CPT(n+1)} = [P_1, P_2, \dots, P_{a+1}]$ as a set of probability estimates for each potential next activity in *NextActs* given *H* so that $A_{CPT(n+1)}$ matches the as-is process as close as possible. $A_{CPT(n+1)}$ can subsequently be used to rank the activities of *NextActs*.

5.2.4. Research Methodology

The strategies described in the next section are the result of a literature review and some creativity. The literature review started from terms like ‘process recommendation’, ‘recommender’ and ‘prediction strategies’. This led to papers on process recommendations and the domain of predictive process monitoring, but also more general and non-process-aware domains. The snowball method was followed to explore the latter, leading to relevant trace clustering techniques, general clustering techniques and string similarity measures. However, most of these were not directly applicable to the process recommendation context. Therefore, we created the heuristic proposed in section 5.2.5.3 to allow these techniques to be applied to this context. Finally, we applied some creativity to find all possible uses for each strategy/measure and added of our own ideas, inspired by related literature. Of course, it should be noted that we excluded a class of potential strategies: black-box strategies (e.g., NN). This is planned as a follow-up comparison between the strategies discussed in this paper and black-box strategies.

We will perform two major experiments using three different healthcare event logs to answer the research question from section 5.2.1. The first experiment will compare the general predictive power of each of the strategies in order to answer RQ1, RQ2 and RQ4. The second experiment will focus

on the evolution of the predictive power of the strategies relative to the number of traces available to learn from. This should provide an answer to RQ3.

5.2.5. Strategies

We have identified 40 strategies to generate next-activity recommendations that all leverage trace similarity in one way or another. Most of these strategies have their roots in the related literature that came up when searching for topics like trace similarity, trace clustering, predictive process monitoring and vector/matrix similarity. We generalized, applied the same idea to other concepts and extrapolated some of those existing strategies to come up with additional strategies. And finally, we used our own creativity and domain knowledge to come up with the four more strategies (i.e., `ActivityWithBefore`s, `DataStateCustomOverlap` and the combined strategies). We purposefully avoided more advanced techniques like neural networks, because of transparency concerns related to their black-box nature. However, in the future we will perform a comparison, including such advanced techniques, to evaluate if they can beat the strategies from this paper and whether the loss of transparency would be worth any potential added precision.

Each strategy was implemented in what we will refer to as a similarity scorer. These scorers take H and a CPT as input and return a list of potential next activities ranked according to their corresponding probability estimates. A scorer can therefore be regarded as the brains of an eventual recommendation system, with the other components being responsible for the integration of the recommendation system in the electronic health record system and for the user interface to convey the results to the users. The scorers are split into three general classes: pre-calculated, positionless and variable-position similarity scorers. Some of the scorers use certain weights and other parameter settings to function properly. The hyperparameter values used in the experiments are given below. Note that these values were determined intuitively before running any experiments (i.e., guesstimates), but special attention was paid to keeping these values consistent across similar strategies. Small scale experiments ex post facto indicated that small variations in the

hyperparameter values had little impact on the relative performance of the different strategies. However, there certainly is room for optimization through tuning.

5.2.5.1. Pre-calculated similarity scorers

These scorers calculate a frequency table once from H and use this table to generate a ranking of potential next activities and a corresponding probability estimate for the CPT . The computation times for these scorers presented in the experiments ignore the computation time of the frequency table, as this is a one-time cost (with a duration of just a couple of milliseconds).

- AbsoluteFrequency (=proportional guessing from Tax et al. (n.d.)): calculates the absolute frequency in H of each activity in $NextActs$.
- ActivityInTraceFrequency (=set abstraction from Schonenberg and Weber (2008)): calculates the frequency of each activity in $NextActs$ as the percentage of traces from H in which it occurs at least once.
- RespondedFrequency (=similar to CFP3 from De Leoni et al. (2016)): calculates for each activity in $NextActs$, and also for the start of trace, the frequency that it is directly followed by a certain other activity from $NextActs$ in the traces of H . The size of the frequency table increases quadratically with the number of activities.
- StepFrequency (=Prefix abstraction from Schonenberg and Weber (2008)): calculates the frequency of each activity in $NextActs$ for each possible position as used in H .

5.2.5.2. Positionless similarity scorers

These scorers calculate the similarity at runtime between the CPT and the completed traces in H , so they do not consider any partial versions of the traces in H .

- IntraTraceFrequency (=multi-set abstraction from Schonenberg and Weber (2008)): calculates a percentage for each activity in $NextActs$ as the number of traces in H that have

more occurrences of the activity than the *CPT*, divided the sum of this number for all activities in *NextActs*.

- IntraTraceFrequencyNotNull: the same as the IntraTraceFrequency scorer, but without considering the end of the trace as a separate activity. The thought process was that adding this extra activity, which occurs eventually in every *HT*, might be detrimental to the performance of the IntraTraceFrequency scorer because the added activity will potentially result in a higher rank and a lower probability estimate of the recommendation of the correct next activity for each prediction point except for the last one of a trace.

5.2.5.3. Variable-position similarity scorers

These scorers calculate the similarity of the *CPT* with a *HT* by calculating the similarity of the *CPT* with all possible *HPTs* of a *HT* that stop in different positions. The similarity is not necessarily calculated for all possible *HPTs* of a *HT*, but rather heuristically. The current position in the *CPT* (i.e., the number of activities already executed for the *CPT*) is used as a starting point, and various positions left and right of the start position are considered until the stop criteria is reached for each direction. This criteria states that the search is stopped if there is no local improvement in the scores of 4 consecutive *HPTs* in that direction. After calculating the similarity score for each trace in *H*, the 100 most similar *HTs* are kept for calculating the probability estimates of the potential next activities. This ‘100’ and ‘4’ are parameters that can be further tuned in the future.

- UniqueActivity: calculates the similarity as the number of unique activities that were executed in both (partial) traces divided by the total number of unique activities executed in the *CPT*, minus a penalty of 0.001 times the number of unique activities that differ between both traces.
- Activity (=activity trace profile from Song et al. (2013) and CFP1 from De Leoni et al. (2016)): calculates the similarity as the weighted sum of two subscores, minus two penalty factors. The first subscore (weight = 0.7857) is the number of unique activities that were

executed in both traces divided by the total number of unique activities executed in the *CPT*. The second subscore (weight = 0.2143) is the complement of the sum of the absolute differences between the number of repeated executions of shared activities of both traces, divided by the total number of activities executed in the *CPT*. The penalty factors are 0.001 times the number of unique activities not shared among both traces and 0.0001 times the number of repeated executions of activities that are not shared.

- ActivityWithBefores: calculates the similarity as the weighted sum of the Activity similarity score (weight = 0.7) and the complement of the sum of the differences in the before-relations of both traces, divided by the sum of the before-relations of the *CPT* (weight = 0.3). The before-relations of a trace consist of a set of activities, with corresponding number of occurrences, for each activity executed in the trace. The list contains all activities that were executed before the corresponding activity was executed. If an activity was executed multiple times, the list of its before-relation will be the sum of the activities executed before each execution.
- UniqueActivityTransition: calculates the similarity as the number of unique activity transitions that are part of both traces, divided by the total number of unique activity transitions in the *CPT*, minus a penalty of 0.001 times the number of unique activity transitions that differ between both traces. The transition from 'null' to the first activity is included.
- ActivityTransition (=transition trace profile from Song et al. (2013)): calculates the similarity as the weighted sum of two subscores, minus two penalty factors. The first subscore (weight = 0.7857) is the number of unique activity transitions that are part of both traces, divided by the total number of unique activity transitions in the *CPT*. The second subscore (weight = 0.2143) is the complement of the sum of the absolute differences between the number of repetitions of activity transitions that are part of both traces, divided by the total number of activity transitions in the *CPT*. The penalty factors are 0.001 times the number of unique

activity transitions not shared among both traces and 0.0001 times the number of repeated activity transitions that are not shared among both traces.

- DataStateCustomOverlap: calculates the similarity as the number of shared currently active data attributes in both traces, divided by the total number of currently active data attributes in the *CPT*, minus a penalty of 0.01 times the number of unique activities that differ between both traces.

We also considered the following general mathematical distance and similarity metrics (Euclidian and Jaccard from Song et al. (2009), Tanimoto from Bajusz et al. (2015) and the others were taken from the SimMetrics Java library, <https://github.com/Simmetrics>, on the condition that they could be easily applied in this context):

- *Dice*: defined as twice the size of the intersection of two sets divided by sum of sizes of each separate set. Also known as the Sørensen index or Sørensen-Dice coefficient.
- *Jaccard*: defined as the size of the intersection of two sets divided by the size of the union of the sets.
- *Overlap Coefficient*: is defined as the size of the intersection of two sets divided by the size of the set that originates from the *CPT*.
- *Tanimoto Coefficient*: defined as the cosine of the angle between two sets expressed as sparse vectors.
- *Block Distance*: defined as the sum of the absolute differences of the Cartesian coordinates of two multisets. Also known as taxicab geometry, rectilinear distance, L1 distance, snake distance, city block distance, Manhattan distance or Manhattan length.
- *Euclidean Distance*: defined as the "ordinary" straight-line distance between two multisets in Euclidean space.
- *Simon White*: same as Dice, but with multisets.

- *Generalized Jaccard*: same as Jaccard, but with multisets.
- *Generalized Overlap Coefficient*: same as Overlap Coefficient, but with multisets.
- *Cosine Similarity*: Tanimoto Coefficient, but with multisets.

These metrics can calculate the distance or similarity between two (multi-)sets. A trace can be converted to such a (multi-)set abstraction by, for example, creating a (multi-)sets of all activities executed in that trace (similar the abstraction used by the the UniqueActivity and Activity scorers). For each of these distance measures and similarity metrics we defined two scorers, and an additional third for all but the Simon White, Generalized Jaccard, Generalized Overlap Coefficient and Cosine Similarity metrics (as these would coincide with the non-multiset versions of the corresponding metric scorers):

- Activity...Scorer: the lists of activities events of the *CPT* and *HPT* are converted to (multi-) sets to calculate the metric.
- Data...Scorer: the lists of data events of the *CPT* and *HPT* are converted to (multi-) sets to calculate the metric.
- DataState...Scorer: the data states of the *CPT* and *HPT* are converted to (multi-) sets to calculate the metric. The data state is the set of data attributes that are active at the current point in the trace. So, it starts from the same (multi-)set as the Data...Scorer but removes all inactive data values.

5.2.5.4. Combined scorers

The scorers above all use an abstraction of just one aspect of the trace. Of course, the eventual goal is to create a scorer that takes multiple aspects into account. As an example of this, we also implemented two scorers that combine several of the aforementioned scorers.

- ActivityWithBeforesAndData: calculates the similarity as the weighted sum of two subscores. The first subscore is the ActivityWithBefores score (weight = 0.6) and the other is the DataStateCustomOverlap score (weight = 0.4).
- ActivityWithBeforesAndDataAndKBs: calculates a score for each potential next activity as a weighted combination of 5 scorers: ActivityWithBefores, DataStateCustomOverlap, RespondedFrequency, StepFrequency and AbsoluteFrequency. The first two are used to calculate a similarity score just like the ActivityWithBeforesAndData scorer, but with slightly different weights (weight of ActivityWithBefores = $0.4/0.59$ and weight of DataStateCustomOverlap = $0.19/0.59$). The final score is then calculated as the sum of ActivityWithBeforesAndData score (weight = 0.59), the RespondedFrequency score (weight = 0.4), the StepFrequency score (weight = 0.009) and the AbsoluteFrequency score (weight = 0.001).

5.2.6. Experiments

We performed two experiments to compare the performance of the different strategies. The first experiment tries to answer RQ1, RQ2 and RQ4 by applying the scorers to three healthcare data sets and comparing the performance of the strategies on different types of data sets. The second experiment is similar, but it varies the size of H to investigate the relation between the predictive power and the size of H to answer RQ3.

5.2.6.1. Experiment 1: comparing the general predictive power over different data sets

5.2.6.1.1. Experimental setup

Three healthcare event logs were used:

- ArmFractures¹⁵: an artificial event log containing 5000 traces (31980 activity and 45147 data events using 9 unique activities and 21 unique data values to form traces of up to 11 activity

¹⁵[https://github.com/stevmert/EventLogs/blob/master/armFractures\(5000\)v2.zip](https://github.com/stevmert/EventLogs/blob/master/armFractures(5000)v2.zip)

events) about patients with suspected arm fractures. It spans parts of the hospital emergency and orthopedic departments and was created via interviews with domain experts.

- Sepsis¹⁶: a real-life event log containing 1050 traces (15214 activity and 32885 data events using 16 unique activities and 214 unique data values to form traces of up to 185 activity events) specifically about sepsis cases that were extracted from the Enterprise Resource Planning system of a hospital.
- EmergencyDepartment: a real-life event log containing 41657 traces (625758 activity and 1350465 data events using 116 unique activities and 951 unique data values to form traces of up to 128 activity events) about the registration, triage, diagnosis and treatment of patients in the medical emergency department of the Belgian AZ Maria Middelaes hospital. We are not at liberty to make this event log public due to a confidentiality agreement.

These event logs contain activity events as well as the relevant data events with the corresponding data values entered by the medical personnel during the execution of the process. The logs can all be categorized as flexible knowledge-intensive processes, but they do differ in how extreme they exhibit these properties. The EmergencyDepartment-log is the most complex data set, yet, it also has the most traces and details for the scores to base their recommendations on.

In this experiment, each scorer was used to calculate recommendations for each prediction point of each trace from the log with all other traces from that log as H . The experiments ran on an Intel Xeon Gold 6140 with 32GB of RAM. The 36-cores were used to calculate the similarity scores of up to 36 traces from H in parallel. The results are evaluated using the trace-wise averages (i.e., for each trace, the average over all of its prediction points is calculated and the average of each of these traces scores is the final average) of four performance measures (see next paragraph), as opposed to the overall averages. Both averages will give similar results, but the overall averages have the undesired effect of giving longer traces a relatively higher weight because they have more prediction

¹⁶Original Sepsis log: <https://data.4tu.nl/repository/uuid:915d2bfb-7e84-49ad-a286-dc35f063a460>

Sepsis log used in this paper (with preprocessed data): <https://github.com/stevmert/EventLogs/blob/master/SepsisCases.zip>

points. The longer traces are typically also harder to predict. Hence, they would skew the results. Trace-wise averages are more representational for general performance.

Three well-known classification performance measures were used to evaluate the performance of the scorers: *accuracy*, *log loss* and *multi-class brier score* (Ferri et al. 2009). These measures quantify the precision of the recommendations (RQ1 and RQ2) by comparing, respectively, the highest ranked recommendation, probability estimate of the highest ranked recommendation and probability estimates of each recommendation to the actual next activity. This gives us a lot of information on the relative predictive power of each strategy. However, the purpose of recommendations in this paper is not merely to predict, but also as a tool to offer guidance to process actors. Therefore, we opted to define an additional performance measure that scores the consistency of the ranking of recommendations (RQ2): the *rank score*. Where the classification performance measures just make a distinction between right and wrong (e.g., ranked first), the rank score makes a distinction between levels of ‘wrong’. For example, ranking the correct next activity as second or 15th makes a big practical difference as process actors cannot be expected to consider much more than a few recommendations at the top.

The rank score is calculated as follows:

$$Rank\ score = (\sum_{i=1}^c f_i r_i)^2$$

With c denoting the number of prediction outcomes/classes (i.e., the size of *NextActs*), f_i representing whether activity i was the correct next activity (i.e., $f_i \in \{0, 1\}$ and $f_i = 1$ iff activity i was the correct next activity) and r_i representing the ranking of potential next activity i .

The ideal rank score is equal to one, as this means that the correct next activity was ranked as first. Sub-optimal rankings will result in a rank score of more than one. The formula is squared so that worse rankings are exponentially penalized. This puts strategies that consistently have the correct next activity at the top of the ranking at an advantage.

5.2.6.1.2. Computational results

The results of experiment 1 are summarized in Table 24 (these results and some additional details are available at <https://github.com/stevmert/recommendationExperiments>). The average rank of the

	Amnifurces					Sepsis					EmergencyDepartment							
	Calculation Time (ms)	Rank	Accuracy	Brier Score	Log Loss	Rank	Calculation Time (ms)	Rank	Accuracy	Brier Score	Log Loss	Rank	Calculation Time (ms)	Rank	Accuracy	Brier Score	Log Loss	Rank
AbsoluteFrequency	0.02	3.64	23.20%	0.0857	2.698	17.90	0.02	4.99	17.51%	0.0529	3.375	36.04	0.12	7.15	18.63%	0.0079	4.015	101.88
ActivityIntraTraceFrequency	0.01	3.82	14.11%	0.0860	2.935	18.63	0.01	5.63	8.38%	0.0534	3.554	40.30	0.11	7.22	8.29%	0.0079	4.110	99.97
StepFrequency	0.01	1.48	74.00%	0.0335	1.004	3.17	0.01	2.47	47.52%	0.0375	2.295	10.15	0.07	4.66	39.17%	0.0058	2.806	64.62
RespondedFrequency	0.01	1.34	80.98%	0.0268	0.750	2.55	0.01	2.28	57.15%	0.0331	1.959	9.80	0.07	3.43	55.43%	0.0047	2.147	44.44
IntraTraceFrequencyNotnull	4.72	2.45	57.25%	0.0785	8.342	12.08	1.84	3.35	43.43%	0.0509	6.531	26.48	106.74	15.04	37.09%	0.0078	7.520	1218.41
IntraTraceFrequency	4.71	2.33	14.11%	0.0737	2.273	6.67	1.86	3.11	8.38%	0.0509	3.088	13.29	106.75	6.42	8.29%	0.0077	3.960	114.66
ActivityBlockDistance	20.37	1.19	87.20%	0.0174	0.461	1.75	3.65	1.79	62.51%	0.0282	1.674	5.37	190.64	4.28	54.74%	0.0048	2.917	124.88
ActivityDice	15.06	1.41	78.86%	0.0319	2.332	3.17	3.53	1.90	58.09%	0.0306	1.828	6.07	248.49	4.45	52.72%	0.0049	3.042	122.90
ActivityEuclideanDistance	21.79	1.19	87.39%	0.0173	0.459	1.74	3.57	1.80	61.83%	0.0287	1.754	5.45	175.71	4.34	54.62%	0.0048	2.938	127.91
ActivityJaccard	15.63	1.41	78.86%	0.0319	2.332	3.17	3.35	1.89	58.00%	0.0306	1.824	6.05	163.56	4.45	52.75%	0.0049	2.829	122.77
ActivityOverlapCoefficient	17.43	1.45	78.43%	0.0322	2.382	3.41	3.37	2.14	50.58%	0.0345	1.980	7.48	204.43	3.92	52.76%	0.0049	2.506	81.94
ActivityTanimotoCoefficient	18.99	1.41	78.86%	0.0319	2.332	3.17	3.24	1.90	57.97%	0.0306	1.829	6.07	205.86	4.45	52.73%	0.0049	2.830	122.73
ActivityCosine	20.26	1.19	87.34%	0.0174	0.461	1.75	3.49	1.78	62.42%	0.0281	1.682	5.33	187.18	4.37	54.43%	0.0048	2.953	128.61
ActivityGeneralizedJaccard	20.28	1.19	87.23%	0.0173	0.461	1.75	3.64	1.78	62.54%	0.0281	1.669	5.33	191.18	4.28	54.74%	0.0048	2.916	124.88
ActivityGeneralizedOverlapCoefficient	20.61	1.25	82.72%	0.0215	0.577	2.02	3.93	2.04	55.48%	0.0327	1.915	7.02	208.11	3.92	54.75%	0.0048	2.576	95.49
ActivitySimonWhite	20.44	1.19	87.20%	0.0174	0.461	1.75	3.83	1.79	62.52%	0.0282	1.674	5.37	203.85	4.28	54.74%	0.0048	2.917	124.88
UniqueActivity	20.78	1.41	78.92%	0.0319	2.332	3.17	3.54	1.88	58.11%	0.0307	1.806	5.93	224.19	4.26	53.26%	0.0048	2.699	111.10
Activity	23.93	1.19	87.42%	0.0172	0.459	1.74	9.48	1.80	62.28%	0.0281	1.754	5.59	215.11	4.14	56.06%	0.0047	2.777	118.37
ActivityUniquTransition	18.72	1.22	86.95%	0.0186	0.851	2.03	4.78	2.03	56.44%	0.0321	2.068	7.22	230.44	4.76	53.87%	0.0049	3.179	153.97
ActivityTransition	19.36	1.18	87.99%	0.0167	0.446	1.72	4.16	2.01	56.69%	0.0321	2.013	6.98	242.56	4.75	54.26%	0.0048	3.213	154.84
ActivityWithBefore	40.52	1.18	88.16%	0.0166	0.444	1.71	11.91	1.82	61.58%	0.0289	1.768	5.61	248.14	4.13	57.32%	0.0046	2.798	123.14
DataDice	20.73	3.47	45.07%	0.0870	18.788	19.00	4.94	4.46	30.02%	0.0547	12.133	34.74	188.10	21.28	27.63%	0.0086	15.701	1211.74
DataJaccard	21.22	3.47	45.02%	0.0870	18.787	19.00	5.00	4.46	30.04%	0.0547	12.141	34.75	181.28	21.27	28.39%	0.0085	15.686	1211.76
DataOverlapCoefficient	21.61	3.49	44.23%	0.0892	19.433	18.99	4.27	3.78	29.91%	0.0525	7.414	25.13	180.89	11.78	37.70%	0.0062	8.099	590.50
DataTanimotoCoefficient	21.57	3.47	45.06%	0.0870	18.783	19.00	4.99	4.48	29.78%	0.0548	12.208	34.85	179.98	28.72	18.27%	0.0099	21.518	1692.17
DataBlockDistance	23.09	3.47	45.07%	0.0870	18.788	19.00	9.26	4.46	30.04%	0.0547	12.138	34.75	336.77	21.28	27.63%	0.0086	15.701	1211.76
DataCosineSimilarity	22.69	3.47	45.06%	0.0870	18.783	19.00	8.55	4.48	29.77%	0.0548	12.206	34.84	331.08	28.72	18.27%	0.0099	21.518	1692.21
DataEuclideanDistance	22.39	3.47	45.06%	0.0870	18.788	19.00	9.16	4.66	28.66%	0.0554	12.069	37.81	635.85	21.46	19.77%	0.0071	14.170	1280.97
DataGeneralizedJaccard	22.31	3.47	45.02%	0.0870	18.787	19.00	5.58	4.46	30.08%	0.0547	12.139	34.74	188.59	21.27	28.39%	0.0085	15.687	1211.79
DataGeneralizedOverlapCoefficient	22.51	3.49	44.23%	0.0892	19.433	18.99	4.66	3.78	29.92%	0.0525	7.413	25.12	198.08	11.78	37.70%	0.0062	8.099	590.49
DataSimonWhite	22.71	3.47	45.07%	0.0870	18.787	19.00	4.47	4.46	30.03%	0.0547	12.136	34.74	199.42	21.28	27.63%	0.0086	15.701	1211.76
DataStateDice	22.75	3.50	45.01%	0.0863	18.671	19.45	3.78	3.37	35.61%	0.0495	5.710	20.12	204.81	10.05	45.20%	0.0057	7.417	501.00
DataStateJaccard	22.88	3.50	44.98%	0.0863	18.670	19.45	4.10	3.37	35.62%	0.0495	5.711	20.12	207.85	10.05	45.24%	0.0057	7.415	501.00
DataStateOverlapCoefficient	23.17	3.52	43.93%	0.0879	18.859	19.50	4.12	3.43	35.70%	0.0501	6.364	21.65	206.55	9.40	45.67%	0.0056	6.763	460.14
DataStateTanimotoCoefficient	23.23	3.50	45.00%	0.0863	18.671	19.45	3.83	3.38	35.57%	0.0496	5.711	20.12	208.01	10.20	44.97%	0.0057	7.529	510.36
DataStateBlockDistance	24.85	3.50	45.01%	0.0863	18.671	19.45	6.52	3.37	35.60%	0.0495	5.710	20.12	426.20	10.05	45.21%	0.0057	7.417	501.00
DataStateEuclideanDistance	25.09	3.50	44.98%	0.0863	18.671	19.45	6.70	3.41	35.39%	0.0498	5.827	20.58	631.70	12.12	39.36%	0.0060	8.232	636.05
DataStateCustomOverlap	23.73	3.50	44.96%	0.0864	18.673	19.45	4.23	3.36	35.59%	0.0497	5.678	19.98	263.53	9.44	46.46%	0.0056	6.898	463.71
ActivityWithBeforeAndData	46.31	1.12	89.71%	0.0133	0.312	1.38	19.87	1.85	60.73%	0.0298	1.855	5.79	364.24	4.28	56.84%	0.0046	2.957	136.28
ActivityWithBeforeAndDataAndKs	25.64	1.13	89.62%	0.0155	0.420	1.42	6.96	1.79	63.85%	0.0291	1.549	5.41	361.76	2.85	60.00%	0.0044	1.941	31.34

Table 24. The results for experiment 1. For each column, the top performer is formatted in *italics* and **bold** and a color-coding was used to represent the best-to-worse ranking (green-yellow-red)

correct next activity is also included, as this is a more relatable than the quite abstract performance measures.

By design, the pre-calculated scorers are the fastest. But it is noteworthy that even the slowest scorer only needed an average of little more than 0.6 seconds of computation time for even the biggest and most complex event log (RQ2).

The color-coding of the table reveals that the results for the different performance measures are clearly correlated. A relatively good result for one measure usually also indicates relatively good performance for the others. In general, the variable-position activity similarity scorers come out on top. Their performance was consistently among the best for all logs. The differences between the different activity similarity scorers are small, but the Activity and ActivityWithBefore scorers consistently perform just slightly better (RQ1). The data and data state variable-position similarity scorers perform poorly across the board. This is not a big surprise, as predicting next activities without knowing what activities have already been performed is inherently more difficult. The gap is smaller for the data state scorers on the data-heavy EmergencyDepartment log. The performance of the positionless similarity scorers is below average. The distinction between the IntraTraceFrequency and IntraTraceFrequencyNotNull scorers was made due to concerns about the impact of adding the end of trace prediction. The results show that these concerns were legitimate. The IntraTraceFrequencyNotNull scorer has a higher accuracy for all logs compared to the IntraTraceFrequency scorer, however, the trade-off is a much higher log loss and rank score.

The performance of the pre-calculated scorers is a bit of a mixed bag. The AbsoluteFrequency and ActivityInTraceFrequency scorers perform poorly, the StepFrequency scorer performs average and the performance of the RespondedFrequency scorer is not that far of the top performers. The latter even generally outperforms the activity similarity scorers on the EmergencyDepartment log. They have similar accuracy and brier scores, but RespondedFrequency scorer has a significantly better log loss and rank score. A closer analysis points the finger to the configuration of the similarity algorithm. We only used the 100 most similar traces from H to calculate the probability estimates

of each potential next activity. Therefore, not all potential next activities are ranked for logs with close-to-or-more activities. If a scorer did not rank the correct next activity, we calculated its rank as the average ranking of all unranked recommendations. For example, the EmergencyDepartment log has 117 potential next activities (=116 activities + 1 end of trace). Suppose a scorer ranked 25 potential next activities, but the correct next activity was not among them. Then the rank of the correct next activity will be calculated as $25 + (117 - 25) / 2 = 71$. Increasing the number of similar traces to consider could improve this, however, this would result in an increase in computation time.

The use of a data state similarity scorer to complement an activity similarity in the ActivityWithBeforesAndData scorer had mixed results. The combined scorer shows a slight improvement in the ArmFractures log compared to just the ActivityWithBefores scorer, but the results are opposite for the other two logs. We suspect that the chosen weights might have been the issue. The weight of the DataStateCustomOverlap similarity score is perhaps a too high (40%). This component has a slightly lower weight (32%), compared to its ActivityWithBefores component, in the slightly better performing ActivityWithBeforesAndDataAndKBs scorer. However, the increase in performance could also come from its RespondedFrequency component. Although, further experiments will be needed to draw any real conclusions about the complementary value of a data similarity scorer, the ActivityWithBeforesAndDataAndKBs scorer clearly combines the best of both worlds (RQ1). It has the predictive power of the similarity scorers and the consistency of the pre-calculated scorers for logs with a high number of activities. And finally, it is also worth noting that the computation time of the combined scorers typically is lower than the sum of the computation times of its component scorers. The explanation can be found in how the component similarity scorers are combined. They do not run in isolation, instead the combination of the similarity components is used directly in the variable-position algorithm. Thus, the convergence of this heuristic can differ from the convergence of the component scorers in isolation, albeit using an inherently slower similarity calculation.

5.2.6.2. Experiment 2: the relation between predictive power and the size of H

5.2.6.2.1. Experimental setup

In this experiment, we created sets containing a varying number of traces to serve as H and separate sets of traces to evaluate the scorers with. The two types of sets were randomly created and never contained overlapping traces. For each log, the size of the set of traces for evaluation was always the same for all sizes of H (500, 100 and 1657, respectively). The experiment ran 3 to 5 times for each size and the average over all runs was calculated. Otherwise, the setup and evaluation criteria were the same as in experiment 1.

5.2.6.2.2. Computational results

The results of this experiment have been summarized in Fig. 31. The less interesting scorers (i.e., those that perform similar or worse than comparable strategies in experiment 1) have been omitted to not overload the graphs (full versions and some additional details are available at <https://github.com/stevmert/recommendationExperiments>).

The trace-wise average of the accuracy, multi-class brier score, log loss and rank score of the similarity scorers all converge very quickly. The performance measures still improve with more and more traces to learn from, but the added value gets smaller and smaller. This corresponds to a logarithmic relation: with a high base number for the accuracy and a fractional base number for the other performance measures. For example, the `ActivityWithBeforeAndDataAndKBs` scorer already reaches an average accuracy of 57.75% with 500 traces in H , and this increases by just 2.52% with 39500 additional traces in H . This is a small improvement in accuracy compared to the increase in computation time by more than a factor 40 (RQ3).

5.2.7. Discussion

The results of the two experiments demonstrate the predictive power of the considered strategies for generating ranked business process recommendation lists as a direct reflection of the experience

Fig. 31. The results for experiment 2

encapsulated in previous executions of the process. We can formulate some conclusions for each research question.

RQ1: *Which prediction strategies perform well for knowledge-intensive and flexible healthcare processes?*

- For logs with a low number of activities, the variable-position activity similarity scorers perform significantly better than the pre-calculated, positionless and variable-position data similarity scorers. Therefore, they show a lot of promise as a high-weight component for combined strategies.
- The performance of the variable-position similarity scorers clearly declines for logs with a high number of activities, because of the limited number of ranked next activities. In this case, the penalty of a correct next activity being unranked weighs heavily on the performance. As we only had one data set with a high number of activities, it is hard to generalize this further or to even use a stricter formulation. The lower performance could also be attributed to the high complexity of the real-life EmergencyDepartment. Future experiments will determine if increasing the number of most similar traces to consider is a good countermeasure and what would be the corresponding impact on the computation times.
- The data and data state variable-position similarity scorers perform unsatisfactory across the board. This was expected, as we envisioned them to be more of an auxiliary and low-weight component for combined scorers. The data state strategies show the most potential.
- Overall, the combined scorers come out on top for all performance measures. Only for the Sepsis log they are beaten by a few variable-position activity similarity scorers on the brier and rank scores, but we suspect this might be due to suboptimal weights and a limited number of traces to learn from (i.e., 1050 compared to 5000 and 41657 for the other event logs). Choosing the subcomponents and weights more carefully, keeping in mind the experimental

results from this paper, could probably also get them to the top for the Sepsis log. Hyperparameter tuning algorithms can be applied to do this automatically.

- The Sepsis log was also used in Tax et al. (n.d.), which compares several other next-element prediction strategies. As explained in section 5.2.2, the goal of these predictions is a bit different to that of this paper. However, the raw predictive power of the techniques can still be compared. Tax et al. use only the brier score to compare the performance of the techniques. Our best strategies outperform the baseline, process mining, machine learning compression, Markov and grammar inference techniques considered in that paper. The artificial neural network and all k-order Markov models (AKON) techniques did have better brier scores, but difference is small (0.0004 and 0.0019, respectively). Perhaps a more meticulous selection of subcomponents for a new combined scorer and some form of hyperparameter tuning would result in similar or even better brier scores. It is worth noting that a better brier score, although they are clearly correlated, does not necessarily mean that the other performance measures used in this paper would agree. For example, the Activity scorer has a better average brier score for the Sepsis log in the first experiment, but a worse average rank, accuracy, log loss and rank score. Comparing the artificial neural network and all k-order Markov models (AKON) techniques using all of our performance measures on a more complex data set like the EmergencyDepartment log will hopefully provide a clearer picture in the future.

RQ2: What kind of precision and consistency can be achieved and in what amount of computation time?

- The achieved performance primarily depends on the characteristics of log. The variable-position activity similarity and combined scorers achieve an accuracy of almost 90% for the ArmFractures log, almost 64% for the Sepsis log and 60% for the EmergencyDepartment log. On average, the correct next activity will be in the top 2 of the variable-position activity similarity and combined scorers' recommendations for the ArmFractures and Sepsis logs, in the top 4 and top 3 for the variable-position activity similarity and combined scorers,

respectively, for the EmergencyDepartment log. The rank scores for the ArmFractures and Sepsis logs show that the variable-position activity similarity scorers perform consistently for logs with relatively few unique activities, but when that number rises they tend to struggle a bit more. The precalculated scorers achieve better performance for the EmergencyDepartment log with its 116 unique activities, while the variable-position combined scorer that combines both takes the crown in consistency.

- The strong performance of the variable-position activity similarity and combined scorers does come at the expense of a higher average computation time. Where the pre-calculated scorers just need to retrieve the ranking from memory, the variable-position similarity scorers must calculate the similarity between the CPT and the traces in H on the spot and subsequently analyze the most similar traces in H for what activity each has as next at its most similar position. However, even the highest average computation time is considerably less than one second. This certainly seems to be acceptable for real-life applications. Moreover, the results of the second experiment show that a relatively small size for H can offer almost the same performance, while severely reducing the computation time.
- The pre-calculated scorers have the lowest computation time, as expected, but even the relatively slow scorers only needed a little more than 0.6 seconds of computation time for even the biggest and most complex event log. This is an excellent result for these unoptimized implementations, albeit with more parallel processing power than might be typically available. It bodes well for its applicability in real-life settings, where a noticeably long computation time could be a source of frustration for the process actors using such a system.

RQ3: How much data is needed to for near-optimal precision?

- Experiment 2 demonstrates that just a couple of hundred traces in H is sufficient for most cases concerning the datasets from this paper or similar datasets. Large sizes for H are only sensible in situations where the maximal performance is required, and processing power is

less of an issue. Of course, the exact number of traces required will strongly depend on the characteristics of process and the quality of the corresponding event log. Larger sizes of H could be needed to achieve near-optimal precision for processes with an even higher complexity or with datasets of lower quality.

RQ4: Which strategies could be useful benchmarks to evaluate more advanced techniques (e.g., neural networks) in the future?

- The RespondedFrequency scorer, and to a lesser degree also the StepFrequency scorer, manages to consistently offer moderate-to-good performance, independent of the complexity and number of activities. The simplicity of the strategy and negligible computation time make it an easy-to-implement (or even fallback) option to provide some guidance. It can also be an excellent benchmark for more advanced strategies and techniques. The performance of the AbsoluteFrequency, ActivityInTraceFrequency, IntraTraceFrequency and IntraTraceFrequencyNotNull scorers is insufficient for these purposes. Their use should be limited to providing relative rankings for otherwise unranked or extremely unlikely next activities, as was the case for the ActivityWithBeforeAndDataAndKBs scorer.

5.2.8. Conclusion and Future Research

The results of the two experiments demonstrate the predictive power of the considered strategies when generating ranked process recommendation lists as a direct reflection of the experience encapsulated in previous executions of the process. Each strategy focuses on specific aspects of the process and they use all sorts of similarity metrics to relate partially executed traces to historic executions.

Although the results vary from log to log and even from performance measure to performance measure, some straightforward trends were revealed. Firstly, there are huge performance differences between the set of simple and pre-calculated strategies. The RespondedFrequency scorer is the clear-cut winner among this group of strategies with its consistent and moderate-to-good overall

performance. This scorer seems like a great choice for a performance benchmark for more advanced techniques. Secondly, the variable-position activity similarity scorers unsurprisingly are top of the class of the scorers that focus on just one aspect of the process. Even for knowledge-intensive processes, previously executed activities seem like the best single-aspect predictors for the future activities, albeit with some meaningful differences between different strategies to represent the previously executed activities. Thirdly, the data and data state variable-position state similarity scorers can be useful as a component that complements another strategy, but not as a standalone scorer, and from this group of strategies, the data strategies that keep a data state are superior to those that do not. Fourthly, the drawbacks of single-aspect strategies can be mitigated by combining strategies. The results of the combined scorers demonstrate that combinations of strategies can outperform the separate building blocks, at a cost of a relatively small increase in computation time. Fifthly, the precision from the simple and unoptimized scorers used in this paper is promising. For the ArmFractures log they could predict the correct next activity with an accuracy of just a hair under 90%. The two real-life logs were clearly more challenging as reflected by a maximal accuracy of 64% and 60%. However, this just reflects the number of times that the correct next activity was predicted as the first ranked recommendation. For offering guidance to the process actors it is still acceptable if the correct next activity was ranked near the top. The best performing scorers were able to rank the correct next activity on average in the top-2 or top-3 of their recommendations. This should ensure that the correct recommendation was at least within reasonable viewing range of the process actors. And finally, the computation time for the different strategies is manageable, even for complex real-life event logs. If necessary, the computation time can be reduced significantly by reducing the size of H (up to a small fraction of all available data) with relatively minimal impact on the overall performance of the scorer.

As part of the future research, we plan to investigate the impact of increasing the number of similar traces used in the variable-position similarity scorers and the use of trace encodings to improve memory usage. For the scorers themselves, we will consider optimizations, hyperparameter

tuning and more combinations of strategies. Advanced techniques like neural networks will also be considered and compared. At a more general level, we would like to apply the techniques to many more event logs. The idea is that this might reveal preferred strategies or techniques for processes on different parts of the flexibility and/or knowledge-intensity spectrum.

RESEARCH CONTRIBUTIONS OF CHAPTER 5

- A generic framework for creating data-aware declarative BPEs
- The proof-of-concept tool support of the framework: DeciClareEngine
- A set of 40 next-activity prediction strategies that can be used as the foundation for a process recommendation generator
- The prediction strategies that perform well for loosely framed KiPs (excluding black-box techniques)
- An idea of the precision and consistency can be achieved by the prediction strategies in a certain amount of computation time
- The amount of data that is required by the prediction strategies for near-optimal precision
- The prediction strategies that could be useful benchmarks to evaluate other strategies, black-box strategies (e.g., neural networks) in particular, in the future

6

Conclusion and Future Research

6.1. RESEARCH RESULTS

The starting point of this dissertation was the observation that current BPM techniques and corresponding BPMSs are not well suited for managing and supporting loosely framed KiPs. They have problems with the complexity that arises from both the loosely framed and knowledge-intensive character of these processes. The result is a very limited adoption of BPM in sectors where KiPs are common (e.g., healthcare, legal profession etc.). Section 1.3 listed three specific objectives that form the foundation of a BPMS for loosely framed KiPs, the development of which was the general objective of this dissertation. The resulting research contributions for each of these objectives, are summarized below.

Objective 1: The development of a declarative process modeling language that suits loosely framed KiPs, which lays the foundation for the Process Designer-component of the BPMS.

- A feature analysis of existing declarative process (Declare, Timed Declare, TBDeclare, ConDec-R, Data-aware Declare, Declare++ and Declare based on LTL-FO) and decision (DMN) modeling languages
- Metamodels of existing declarative languages (Declare and ConDec-R)
- The DeciClare declarative process and decision modeling language, with integrated functional, control-flow, data and resource perspectives
- The DeciClare Process Designer-component (textual)

The core problem is a mismatch between the characteristics of loosely framed KiPs and the leading modeling paradigm, which takes on an imperative viewpoint. The imperative modeling paradigm thrives in settings with little variation but degenerates to useless spaghetti models in settings where executional flexibility is crucial. The stakeholders of loosely framed processes are left with the impossible choice to either artificially reduce the executional freedom they need to operate properly in order to accommodate mainstream BPM techniques, or just reject the adoption of most

of these techniques and perpetuate the current overextension of knowledge workers that lacks transparency and often also efficiency. Therefore, a paradigm shift is essential to provide them with the process support they need. The declarative modeling paradigm has been proposed before as a potential solution. It offers flexibility by design, without the requirement to enumerate every possible valid process variation explicitly. With the addition of resource and data perspectives, a declarative modeling language can also cater to the knowledge-intensive characteristics. The former is needed to model the multidisciplinary aspects of many knowledge-intensive processes, while the latter enables modeling the decisions that govern the choice for a specific process variation. The disadvantage of the declarative paradigm compared to imperative paradigm comes down to human understandability. Declarative models have less or even no explicit flow, and therefore are inherently difficult for humans to comprehend easily. Declarative languages like DCR Graphs try to mitigate this problem by limiting the number of constraint types and providing a clear visual syntax. However, in the scope of this dissertation, the models are targeted more towards automated interpretation by computers, and so, this disadvantage is less relevant.

For the fulfillment of the objective, we first listed the requirements for a modeling language for loosely framed KiPs. Based on these requirements, we analyzed the features of existing declarative process modeling languages. The DeciClare modeling language was subsequently developed by incorporating features of the existing languages and several new features to satisfy the requirements. The domain appropriateness was evaluated by domain experts to verify a good match between the characteristics of loosely framed KiPs and the newly developed modeling language.

Objective 2: The development of a method for process and decision discovery of loosely framed KiPs for the Process Discovery-component of the BPMS, and a corresponding mining technique that can mine a process and decision model from an event log.

- The DeciClareMiner process and decision mining technique and corresponding tool

- A method for process and decision discovery of loosely framed KiPs for the Process Discovery-component that uses the DeciClareMiner process mining technique
- A case study evaluation at the emergency medicine department of AZ Maria Middelaers, Ghent, Belgium

Process discovery is traditionally done by way of interviews with process actors and other domain experts. This is straightforward when there is little knowledge to extract, but otherwise can quickly become a very difficult, effort-intensive, time-consuming and error-prone task for both the modeler and the process actors. The manual discovery of loosely framed KiPs certainly falls in the latter category. Most of the process knowledge is tacit and scattered over the multi-disciplinary team of process actors. Process mining techniques offers an alternative way of extracting some of this knowledge, if enough digital process data is available and that data is of sufficient quality. Therefore, we developed a process and decision mining technique, DeciClareMiner, that can discover a model written in the declarative modeling language that was the result of objective 1. It consists of two phases. The first phase consists of a complete search algorithm inspired by the Apriori association rule mining algorithm, while the second phase uses a heuristic belonging to the class of genetic algorithms. This difference in approach for each phase is rooted in the assumption that in the context of loosely framed KiPs, the number of possible activities will be much lower than the number of combinations of available data elements on which decisions are based. This means that using a complete-search algorithm for the first phase is acceptable, while for the second phase, a heuristic is more appropriate. Additionally, the search space of the second phase algorithm was significantly reduced by efficiently making use of a subset of constraints that was discarded, but stored, during the first phase (i.e., minimal rules). The output is a model of the process that describes the given process both in terms of the process control-flow and the decisions that govern the control-flow.

The literature on process mining often assumes that a satisfactory event log is available, and that the execution of the process mining technique is the only step of a discovery project. However, this

is not a good reflection of reality. The process mining techniques are just part of one step of a more general method to discover processes. Two methodologies to perform process discovery have been published, but they presume the use of an imperative process mining technique. So, a new method was developed that consists of gathering the necessary data, structuring that data, preparing the resulting event log, mining a declarative model and evaluating that model. These steps should be repeated iteratively until a satisfactory model is discovered. Of course, the method centers around the use of the DeciClareMiner technique, but the quality of the output depends at least as much on the other steps, if not more, due to importance of data quantity and quality. During the development of the method, we simultaneously applied it to real data from the emergency care process that takes place in the medical emergency department of a hospital. The result was a process and decision model that successfully transformed some of the tacit process knowledge to explicit knowledge. Instead of numerous and time-consuming interviews with the process actors, it merely required an occasional interview with one domain expert to evaluate the discovered model for each iteration.

Objective 3: The development of a BPE- and Process Analytics-component for the BPMS that can offer support and guidance to the process actors during process execution.

- The DeciClare BPE-component that supports
 - Process execution
 - Runtime conformance checking
 - Process replay capabilities
- A recommendation generator for the BPE-component that provides guidance to the users and is the first part of the Process Analytics-component

The development of a process and decision modeling language and mining technique for loosely framed KiPs fulfills the first two objectives. The declarative paradigm of the resulting models represents many process variations implicitly, which is good for modeling and discovery but

difficult for humans to interpret due to the lack of explicit process paths to follow. Computers are much better at interpreting declarative models. Therefore, an automated system to support users that need to execute a declarative model would be very helpful. This is typically a job for a BPMS, and more specifically, the BPE-component of a BPMS. A BPE automatically interprets a given model and presents the user only with the actions that are allowed by the model in the current phase of the execution of a process instance. Traditional BPEs execute imperative process models that often consist of just one or two explicit process paths. They are not capable of executing a declarative process model, hence, a new type of BPE needed to be developed. Additionally, the new BPE also needed to be able to handle the extensive data perspective of the declarative process and decision modeling language that resulted from objective 1. The problem was tackled by creating a BPE that categorizes the activities according to whether or not they are allowed as next activity by the given model in the current context of the running process instance at hand both from a control-flow and data perspective. The users must then merely decide between the activities that are allowed by the model, keeping in mind any potential consequences that are noted by the engine (e.g., an additional activity must to be executed at a later time). The reasons why other activities are prohibited by the model are also provided so that the users can better understand the calculated categorization of activities. The result is that users are relieved of the difficult task of interpreting the model and can focus their full attention on the decision-making tasks, for which they are presented with all information available in the given model. However, as the processes are highly flexible, the model will often allow multiple activities to be performed next. Therefore, we investigated ways to provide additional guidance on which of the available next activities are preferable or typical in the current context of the running instance. A solution was devised that ranks the available next activities as process recommendations based on their frequency of occurring as the next activity in the most similar previously completed process instances. The rank and probability estimation of the process recommendations convey to the users how current and past knowledge workers of the organization would proceed, and hence, offer a direct reflection of the combined past and present experience in

the organization. Many strategies from related literature to implement this type of recommendation generator, complemented with some novel strategies, were evaluated and compared. Eventually, this recommendation generator will be part of the Process Analytics component of the BPMS for loosely frame KiPs, along with other algorithms for different analytics tasks.

6.2. RESEARCH RELEVANCE AND IMPLICATIONS

Although many loosely framed KiPs are an integral part of our daily life, the process management support for these types of processes is severely lacking. In this dissertation we developed the fundamental building blocks needed to provide a basic level of process management, similar to how this is already available for other types of processes. The implication of the dissertation can be divided along two interest groups: researchers and process stakeholders.

6.2.1. Implications for researchers

First off, the developed components for a BPMS for loosely framed KiPs can be the foundation for new decision support system research, as alternative to current lines of research into guidelines implementation and expert systems. The application of BPM in this context has not yet been thoroughly researched, both from a theoretical and empirical viewpoint. Secondly, new process monitoring, conformance-checking and simulation research avenues can be explored due to the higher expressive power of the developed process modeling language compared to previously available languages. And finally, a lot of research has now been enabled towards finding the best ways to incorporate these systems and techniques into practice. The foundation for the technical side has been provided, but the adoption of the resulting systems and techniques in practice requires more research into when, what and how to present them to the process stakeholders.

6.2.2. Implications for process stakeholders

The managers and other stakeholders of loosely framed KiPs are now often left in the dark about what is happening and why. The results of this dissertation can be used to transform a lot the tacit

knowledge and experience into explicit knowledge that can be analyzed to gain valuable insight into these processes. The implication of this are plentiful:

- Knowledge workers are relieved of some of the more time-consuming task of having to reinvent the wheel over and over to determine a path towards a satisfactory process ending. They get the necessary support for applying all the rules, guidelines and norms of the organization and can fully focus on the execution of the actual process activities and the crucial decision-making tasks along the way.
- Having explicit rules for the occurrence and sequencing of activities and the corresponding decision logic enables available deeper analysis and optimization of the working habits of knowledge workers. The decision logic in the models can also be compared to the available guidelines (e.g., Are the guidelines properly applied? If not, what are the characteristic and decision criteria of the cases in which they diverge from the guidelines? Are these justifiable?).
- The explicit knowledge extracted by the proposed techniques can be used as an explicit justification as to why certain activities are performed in specific cases and others are not.
- The service offered as a result of the execution of the loosely framed KiPs can potentially be made to be more uniform. The current lack of transparency and support makes it difficult for the knowledge workers to gauge how their own habits compare to those of their colleagues. By making the process more transparent and providing insight to how the knowledge workers' habits stack up to those of their colleagues, we would assume that this would lead to a more uniform way of executing the process.
- The mined models can be used for evidence farming (Hay et al. 2008). This is the practice of posteriori analysis of data to find new insights without setting up case-control studies. Evidence farming can be used as an alternative or, even better, as a supplement to evidence-based knowledge gathering.

- New findings and working habits can be communicated and enforced directly through the BPMS by adding them to the process model.
- Every organization must deal with employee churn, but for loosely framed KiPs it can be even more devastating. An experienced knowledge worker cannot be replaced easily, and his/her knowledge and experience are typically lost. Even when an experienced replacement can be found, there is no way to tell if the replacement adequately compensates the loss. The proposed BPMS can make sure that not all of the knowledge and experience of the departing employee is lost, because some of the knowledge and all of the experiences and tendencies of the departing employee during his/her stay at the organization were captured in the data of the system. The knowledge captured in the data will have been made explicit in the model used by the BPMS and the experiences and tendencies have been made available through the recommendation generator. This makes the loss a bit less devastating, because the models can be used to educate new employees. Additionally, the experiences and tendencies of current and past colleagues made available by the recommendation generator will provide the necessary support to bring them up to speed more quickly.
- The educational value for new employees can also be generalized to improve educational system that supplies future knowledge workers. The models and recommendations could be used in direct teaching methods, but also to provide realistic replay and simulation capabilities that better prepare them for the job and support them when they are eventually deployed.

Many of the above implications require the input data for the discovery and recommendations to reach a certain level of data quality and detail. For the former we can also make a distinction between the level of detail of the activity definitions (e.g., ‘request lab-tests’ vs. ‘request urine test for nickel poisoning’) and of the contextual data (‘pain’ vs. ‘pain in shoulder’ vs. ‘pain in shoulder when moving the arm upwards’), while the latter relates to all data in general. The required levels are hard to quantify, as they depend on the use case and inherent complexity of the underlying process. The

more complex the patterns and decisions that need to be identified, the higher the levels of data quality and detail that must be achieved. In my opinion, the data quality and level of detail in the activity definitions should be prioritized over the level of detail of the contextual data, because the data quality directly relates to the confidence one should have in the applications and the activity definitions directly relates to the type of knowledge that can be made explicit. These both heavily influence the usefulness, user-friendliness and amount of guidance any the applications can offer.

On the flip side, increasing the level of detail in the input data will incur a cost computationally: the search space that the process mining tool must cover increases exponentially as it grows, the BPE will be slower to categorize the potential next activities due to more complex models and the recommendations generator will be slower to provide recommendations and predictions due to more data needing to be compared. Therefore, this important tradeoff between speed and precision must certainly be carefully considered. Note that increasing the data quality does not necessarily come at a computational cost. This is another reason to prioritize the data quality in these kinds of projects.

6.2.3. Generalizability to other domains

This dissertation focusses heavily on the healthcare domain, because it is perhaps the most extreme example, but loosely framed KiPs occur in many other branches of the service industry. Consider the following examples:

- Legal profession: liability lawyers are hired by individuals or companies to represent them in a legal battle over liability in some event. After reviewing the case at hand and speaking with the client, they must establish a legal strategy by making an interpretation of the law. Based on the strategy they go into a negotiation with the legal teams of the other parties in the case, request evidence to be turned over, review evidence, request different types of expert testimony, search for legal precedent, etc. Additionally, they must respect the procedural rules of the specific court in front of which they are to appear (i.e., timely send documents, requests and notifications to court officials and other parties). If no agreement was reached,

the case goes to trial. During the trial, all parties plead their cases to a judge, who will eventually hand out a verdict. A verdict does not necessarily mean the end of the case, as any one of the involved parties could decide to fight the decision based on the available grounds for appeal of the specific court that gave the verdict. This could start a new round in the legal battle with different procedures depending on the new court to which the case would be moved.

In comparison with the healthcare cases from this dissertation, the activities before a trial are comparable to diagnostic tests, a verdict to a preliminary diagnosis and an appeal to a preliminary diagnosis turning out to be wrong. A trial is more like surgery, because what goes on during is beyond the scope of the system proposed in this dissertation. Therefore, it should be treated like an atomic activity. The abundance of legal strategies, procedures, interpretations of the law and legal precedents leads to much variation and scattered knowledge, while the execution of the process depends heavily decisions made by the lawyers involved based their knowledge and experience.

- Customer services: customer service agents are contacted by customers with a complaint. The subject of a case can be just about anything related to the products or services offered by the company for which they provide customer service. The customer service agent must first gather information about the problem by listening to the customer and asking the right questions. With this information, the agent can start examining the problem and possibly order internal investigations of the malfunctioning product or service. Based on the findings, the agent can consider different ways to resolve the problem or at least appease the customer. The agent might also have to ask permission from his/her manager for some of the resolution options.

The job of a customer service agent shows a lot of similarity to the healthcare examples presented in this dissertation. They also must diagnose a problem based on the symptoms as described by the customer, investigate the problem with diagnostic tests or support from the

experts from within the company, and propose solutions to resolve the problems (i.e., a treatment). The difference mostly relates to the level of flexibility and decision autonomy permitted to the customer service agents. Their decision autonomy and options available for investigation and resolution are often limited in real-life, arguably to the detriment of all stakeholders (i.e., less satisfied customers lead to reputation damage and lost revenue for the company). In my own experience, smaller companies often permit their agents more decision autonomy, which they use to provide a better and more personalized service.

- Air traffic control: air traffic controllers are responsible for safe and efficient incoming, outgoing and overflying air traffic in a controlled airspace. They must communicate with pilots, airport personnel and other air traffic controllers, manage the four-dimensional movement of the air traffic, constantly determine the priority of events, make safety adjustments were needed and deal with all sorts of emergency situations.

Although it is harder to draw a direct parallel with the healthcare cases of this dissertation, this is certainly a loosely framed KiP. The controller must constantly use his/her knowledge and experience to maintain an overview of the situation, assess threats, make split-second decisions and intervene timely when necessary, while the unique context of each situation and the multitude of other actors and possible actions results in a high need for run-time flexibility.

Each of these examples display the characteristics of loosely framed KiPs and the same external pressure to provide more service in less time. The differences between these other examples and the healthcare cases used throughout this this dissertation relate predominantly to the interactions with the clients and how stationary that the knowledge workers are during the execution of these processes. Lawyer often have with little contact with their clients, just enough to keep them up-to-date. Customer service agents and air traffic controllers, in contrast, are in constant audio communication with their clients (i.e., customers and pilots, respectively). And while physicians are typically running around and having direct contact with the clients (i.e., patients), lawyers, customer

service agents and air traffic controllers generally do most of their work on their computer at their desk. The fact that the knowledge workers in these other domains are mostly stationary, should make it easier to capture the process execution data. For customer service agents and air traffic controllers this is certainly the case, while lawyers are starting to catch up by utilizing more and more digital technology. And so, the benefits of the system proposed in this dissertation should also hold true for other domains that house loosely framed KiPs.

6.3. RESEARCH LIMITATIONS AND FUTURE RESEARCH

In this dissertation, several proof-of-concept tools and techniques were proposed. A proof-of-concept demonstrates the core principles of a design artifact. Therefore, it does not require every feature or aspect that might be needed for general usage. The missing features and aspects of the proposed artifacts are listed below, as well as some additionally interesting avenues for future research.

- The DeciClare process modeling languages has an expressive metamodel and corresponding set of constraint templates. However, the current visualization is merely textual. A real visual syntax is yet to be developed and could potentially make it easier for users to quickly understand a model or parts of it.
- The DeciClareMiner process discovery technique can already handle the 9 most frequently used templates (out of 36). However, the support will be further expanded. Additionally, the feasibility of mining information and resources templates from process data will to be investigated. This will require some resource-related data that is not yet available in common event logs, but perhaps the logs can be enriched with this data.
- The models that DeciClareMiner returns when the confidence is set to less than 100% can contain inconsistencies. This is because a rule that conforms to 90 out of 100 instances in which it is activated (e.g., do activity 1 next), can be inconsistent with a rule that conforms perfectly (or at least more than the conformance threshold) to the 10 missing instances (e.g.,

do activity 2 next). For these 10 instances, the model will demand two contradictory actions. Techniques to resolve issues like this have been proposed for Declare (Di Ciccio et al. 2017), but they will need to be adapted to become useful for DeciClare due to its support for (de-) activation decisions.

- The inclusion of more advanced text mining and natural language processing techniques, like TiMBL, in the ‘Log preparation’-step of the method for process and decision discovery of loosely framed KiPs will be further investigated. This has the potential to improve the data quality of the resulting event log, which in turn can improve the discovered process and decision models.
- The BPE-component, DeciClareEngine, already supports 17 of the 36 constraint templates of DeciClare, but the support is not yet complete. Support for the remaining templates will be added in the future, as well as for deactivation decisions. However, not all of these templates can be enforced by a BPE (e.g., ‘at least 5 nurses need to be available’). Additionally, methods for partial live- or deadlock detection or prevention will be investigated. At a practical level, support will also be added for dynamic resource availabilities and parallel execution of activities, while more efficient validation techniques will be investigated to speed up the engine. And finally, more advanced simulation capabilities will be added to the engine on top of the current replay capabilities of historic instances.
- The current recommendation generator only makes suggestions for the next activity to be executed. We will investigate whether it would be beneficial to also include sequences of multiple activities to be executed next. This could be because of standard procedure, as part of a medical guideline or just because the choice for the first activity of the sequence predetermines the others. This has the potential to make of the consequences of a decision more explicit. Those consequences will also be made clearer by providing the user with an overview of the differences in the impact of each recommendation on the remainder the

current instance (e.g., activity X is prohibited in case recommendation 1 is selected but selecting recommendation 2 would mean activity X is available afterwards).

- The techniques behind the recommendation generator can also be used to provide the user with predictions of key performance indicators (KPIs) and other process characteristics (e.g., remaining time, urgency cost, etc.). This predictive component will become the process analytics component of the BPMS for loosely framed KiPs.
- The decision modeling aspect of DeciClare focusses on the specific combinations of data values that lead to certain behavior (e.g., if a patient has a suspected lower arm fracture then a scan needs to be made), but these data values are just the consequences of an actual decision rationale that was applied (e.g., a scan is needed to check if there is indeed a fracture and, if so, to determine the position of the fractured bone endings). The decision rationale is certainly more valuable knowledge. During the first stages of this research project, it quickly became clear that we lacked the means to make the decision rationales explicit without requiring a huge time and effort investment of the knowledge workers. The only options to gather some data on decision rationales are through interviews or by asking the knowledge workers to think aloud during their daily tasks and recording this. The former is often inefficient and incomplete, while the latter is very invasive and difficult to use on a large scale. So, we focused on the data that was already being captured (i.e., the process data in the information systems of an organization). However, this data is not sufficiently rich to reconstruct the decision rationales. The quality of a system as envisioned in this dissertation could certainly benefit if users would be motivated to include the rationale in the data, as complementary data, so future work should certainly explore this path.

References

- Van Der Aa, H., Leopold, H., and Batoulis, K. 2015. "Integrated Process and Decision Modeling for Data-Driven Processes," in *BPM'15 Workshops, DeMiMoP'15. LNBIP, Vol. 256*, Innsbruck, Austria: Springer, pp. 405–417.
- van der Aalst, W. M. P. 1998. "Three Good Reasons for Using a Petri-Net-Based Workflow Management System," in *Information and Process Integration in Enterprises. SECS, Vol 428*, T. Wakayama, S. Kannapan, C. M. Khoong, S. Navathe, and J. Yates (eds.), Springer, pp. 161–182. (https://doi.org/10.1007/978-1-4615-5499-8_10).
- van der Aalst, W. M. P. 2009. "Process-Aware Information Systems: Lessons to Be Learned from Process Mining," *Transactions on Petri Nets and Other Models of Concurrency II*, pp. 1–26. (https://doi.org/10.1007/978-3-642-00899-3_1).
- van der Aalst, W. M. P. 2011a. *Process Mining*, Berlin, Heidelberg: Springer Berlin Heidelberg. (<https://doi.org/10.1007/978-3-642-19345-3>).
- van der Aalst, W. M. P. 2011b. "Workflow Resource Patterns." (<http://www.workflowpatterns.com/patterns/resource/>, accessed October 6, 2015).
- van der Aalst, W. M. P. 2011c. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*, Springer.
- van der Aalst, W. M. P. 2013. "Business Process Management: A Comprehensive Survey," *ISRN Software Engineering* (2013), pp. 1–37. (<https://doi.org/10.1155/2013/507984>).
- van der Aalst, W. M. P., Adriansyah, A., Alves de Medeiros, A. K., Arcieri, F., Baier, T., Blickle, T., Jagadeesh Chandra Bose, R. P., van den Brand, P., Brandtjen, R., Buijs, J., Burattin, A., Carmona, J., Castellanos, M., Claes, J., Cook, J., Costantini, N., Curbera, F., Damiani, E., de Leoni, M., Delias, P., van Dongen, B. F., Dumas, M., Dustdar, S., Fahland, D., Ferreira, D. R., Gaaloul, W., van Geffen, F., Goel, S., Günther, C. W., Guzzo, A., Harmon, P., ter Hofstede, A. H. M., Hoogland, J., Ingvaldsen, J. E., Kato, K., Kuhn, R., Kumar, A., La Rosa, M., Maggi, F. M., Malerba, D., Mans, R. S., Manuel, A., McCreesh, M., Mello, P., Mendling, J., Montali, M., Motahari-nezhad, H. R., zur Muehlen, M., Munoz-Gama, J., Pontieri, L., Ribeiro, J., Rozinat, A., Pérez, H. S., Pérez, R. S., Sepúlveda, M., Sinur, J., Soffer, P., Song, M., Sperduti, A., Stilo, G., Stoel, C., Swenson, K. D., Talamo, M., Tan, W., Turner, C., Vanthienen, J., Varvaressos, G., Verbeek, E., Verdonk, M., Vigo, R., Wang, J., Weber, B., Weidlich, M., Weijters, T., Wen, L., Westergaard, M., and Wynn, M. 2011. "Process Mining Manifesto," in *Business Process Management Workshops. BPM'11. LNBIP, Vol. 99*, Springer, pp. 169–194. (<https://doi.org/10.1016/j.is.2011.10.006>).
- van der Aalst, W. M. P., Pesic, M., and Schonenberg, H. 2009. "Declarative Workflows: Balancing between Flexibility and Support," *Computer Science - Research and Development* (23:2), pp. 99–113. (<https://doi.org/10.1007/s00450-009-0057-9>).
- Ackermann, L., Schöning, S., Petter, S., Schützenmeier, N., and Jablonski, S. 2018. "Execution of Multi-Perspective Declarative Process Models," in *On the Move to Meaningful Internet Systems. OTM 2018. LNCS, Vol. 11230*, H. Panetto, C. Ebruyne, H. Proper, C. Ardagna, D. Roman, and R. Meersman (eds.), Springer, pp. 154–172. (https://doi.org/10.1007/978-3-030-02671-4_9).
- Agrawal, R., and Srikant, R. 1994. "Fast Algorithms for Mining Association Rules," in *VLDB*, pp. 487–499. (<https://doi.org/10.1.1.40.7506>).
- Aguilar, E. R., García, F., Ruiz, F., Piattini, M., Calahorra, L., García, M., and Martin, R. 2008. "Process Modeling of the Health Sector Using BPMN: A Case Study," in *HEALTHINF 2008* (Vol. 2), pp. 173–178.
- Amorim Lopes, M., Soares, C., Almeida, Á., and Almada-Lobo, B. 2016. "Comparing Comparables: An Approach to Accurate Cross-Country Comparisons of Health Systems for Effective Healthcare Planning and Policy Guidance," *Health Systems* (5:3), Nature Publishing Group, pp. 192–212. (<https://doi.org/10.1057/s41306-016-0001-8>).
- Andersen, S. N., and Broberg, O. 2017. "A Framework of Knowledge Creation Processes in Participatory Simulation of Hospital Work Systems," *Ergonomics* (60:4), pp. 487–503. (<https://doi.org/10.1080/00140139.2016.1212999>).
- Anyanwu, K., Sheth, A. P., Cardoso, J., Miller, J. A., and Kochut, K. J. 2003. "Healthcare Enterprise Process

- Development and Integration,” *Journal of Research and Practice in Information Technology* (35:September), pp. 83–98.
- Bajusz, D., Rácz, A., and Héberger, K. 2015. “Why Is Tanimoto Index an Appropriate Choice for Fingerprint-Based Similarity Calculations?,” *Journal of Cheminformatics* (7:1), Journal of Cheminformatics, pp. 1–13. (<https://doi.org/10.1186/s13321-015-0069-3>).
- Baker, K., Dunwoodie, E., Jones, R. G., Newsham, A., Johnson, O., Price, C. P., Wolstenholme, J., Leal, J., McGinley, P., Twelves, C., and Hall, G. 2017. “Process Mining Routinely Collected Electronic Health Records to Define Real-Life Clinical Pathways during Chemotherapy,” *International Journal of Medical Informatics* (103), Elsevier Ireland Ltd, pp. 32–41. (<https://doi.org/10.1016/j.ijmedinf.2017.03.011>).
- Barba, I., Lanz, A., Weber, B., Reichert, M., and Del Valle, C. 2012. “Optimized Time Management for Declarative Workflows,” *Lecture Notes in Business Information Processing* (113 LNBIP), pp. 195–210. (https://doi.org/10.1007/978-3-642-31072-0_14).
- Barba, I., Weber, B., Del Valle, C., and Jiménez-Ramírez, A. 2013. “User Recommendations for the Optimized Execution of Business Processes,” *Data & Knowledge Engineering* (86), pp. 61–84. (<https://doi.org/10.1016/j.datak.2013.01.004>).
- Basole, R. C., Braunstein, M. L., Kumar, V., Park, H., Kahng, M., Chau, D. H., Tamersoy, A., Hirsh, D. A., Serban, N., Bost, J., Lesnick, B., Schissel, B. L., and Thompson, M. 2015. “Understanding Variations in Pediatric Asthma Care Processes in the Emergency Department Using Visual Analytics,” *Journal of the American Medical Informatics Association* (22:2), pp. 318–323. (<https://doi.org/10.1093/jamia/ocu016>).
- Benevides, A. B., Guizzardi, G., Braga, B. F. B., and Almeida, J. P. A. 2010. “Validating Modal Aspects of OntoUML Conceptual Models Using Automatically Generated Visual Word Structures,” *Journal of Universal Computer Science* (16:20), pp. 2904–2933. (<https://doi.org/10.1.1.430.2947>).
- Bernardi, M. L., Cimitile, M., Di Lucca, G., and Maggi, F. M. 2012. “Using Declarative Workflow Languages to Develop Process-Centric Web Applications,” in *Proceedings of IEEE International EDOC 2012 Workshops*, Beijing, China, pp. 56–65. (<https://doi.org/10.1109/EDOCW.2012.17>).
- Birkmeyer, J. D., Finlayson, E. V. A., and Birkmeyer, C. M. 2001. “Volume Standards for High-Risk Surgical Procedures: Potential Benefits of the Leapfrog Initiative,” *Surgery* (130:3), pp. 415–422. (<https://doi.org/10.1067/msy.2001.117139>).
- De Bleser, L., Depreitere, R., De Waele, K., Vanhaecht, K., Vlayen, J., and Sermeus, W. 2006. “Defining Pathways,” *Journal of Nursing Management* (14:7), pp. 553–563. (<https://doi.org/10.1111/j.1365-2934.2006.00702.x>).
- Borrego, D., and Barba, I. 2014. “Conformance Checking and Diagnosis for Declarative Business Process Models in Data-Aware Scenarios,” *Expert Systems with Applications* (41:11), Elsevier, pp. 5340–5352. (<https://doi.org/10.1016/j.eswa.2014.03.010>).
- Braga, B. F. B., Almeida, J. P. A., Guizzardi, G., and Benevides, A. B. 2010. “Transforming OntoUML into Alloy: Towards Conceptual Model Validation Using a Lightweight Formal Method,” *Innovations in Systems and Software Engineering* (6:1), pp. 55–63. (<https://doi.org/10.1007/s11334-009-0120-5>).
- Braun, R., Schlieter, H., Burwitz, M., and Esswein, W. 2014. “BPMN4CP: Design and Implementation of a BPMN Extension for Clinical Pathways,” in *IEEE BIBM 2014*, pp. 9–16. (<https://doi.org/10.1109/BIBM.2014.6999261>).
- vanden Broucke, S. K. L. M., Vanthienen, J., and Baesens, B. 2014. “Declarative Process Discovery with Evolutionary Computing,” *Proceedings of the 2014 IEEE Congress on Evolutionary Computation, CEC 2014*, pp. 2412–2419. (<https://doi.org/10.1109/CEC.2014.6900293>).
- Burattin, A., Cimitile, M., Maggi, F. M., and Sperduti, A. 2015. “Online Discovery of Declarative Process Models from Event Streams,” *IEEE Transactions on Services Computing* (8:6), IEEE, pp. 833–846. (<https://doi.org/10.1109/TSC.2015.2459703>).
- Burattin, A., Maggi, F. M., van der Aalst, W. M. P., and Sperduti, A. 2012. “Techniques for a Posteriori Analysis of Declarative Processes,” in *Proceedings of IEEE International EDOC 2012*, Beijing, China, pp. 41–50. (<https://doi.org/10.1109/EDOC.2012.15>).
- Burattin, A., Maggi, F. M., and Sperduti, A. 2016. “Conformance Checking Based on Multi-Perspective Declarative Process Models,” *Expert Systems with Applications* (65), Elsevier Ltd, pp. 194–211. (<https://doi.org/10.1016/j.eswa.2016.08.040>).

- Campbell, H., Hotchkiss, R., Bradshaw, N., and Porteous, M. 1998. "Integrated Care Pathways," *BMJ* (316:133). (<https://doi.org/10.1136/bmj.316.7125.133>).
- Caron, F., Vanthienen, J., and Baesens, B. 2013. "Comprehensive Rule-Based Compliance Checking and Risk Management with Process Mining," *Decision Support Systems* (54:3), Elsevier B.V., pp. 1357–1369. (<https://doi.org/10.1016/j.dss.2012.12.012>).
- De Carvalho, R. M., Silva, N. C., Lima, R. M. F., and Cornélio, M. L. 2013. "ReFlex: An Efficient Graph-Based Rule Engine to Execute Declarative Processes," *Proceedings - 2013 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2013*, pp. 1379–1384. (<https://doi.org/10.1109/SMC.2013.238>).
- Di Ciccio, C., Maggi, F. M., and Mendling, J. 2016. "Efficient Discovery of Target-Branched Declare Constraints," *Information Systems* (56), Elsevier, pp. 258–283. (<https://doi.org/10.1016/j.is.2015.06.009>).
- Di Ciccio, C., Maggi, F. M., Montali, M., and Mendling, J. 2015. "Ensuring Model Consistency in Declarative Process Discovery," in *BPM'15. LNCS, Vol. 9253*, Innsbruck, Austria: Springer, pp. 144–159. (https://doi.org/10.1007/978-3-319-23063-4_9).
- Di Ciccio, C., Maggi, F. M., Montali, M., and Mendling, J. 2017. "Resolving Inconsistencies and Redundancies in Declarative Process Models," *Information Systems* (64), Elsevier Ltd, pp. 425–446. (<https://doi.org/10.1016/j.is.2016.09.005>).
- Di Ciccio, C., Marrella, A., and Russo, A. 2015. "Knowledge-Intensive Processes: Characteristics, Requirements and Analysis of Contemporary Approaches," *Journal on Data Semantics* (4:1), pp. 29–57. (<https://doi.org/10.1007/s13740-014-0038-4>).
- Di Ciccio, C., and Mecella, M. 2015. "On the Discovery of Declarative Control Flows for Artful Processes," *ACM Trans. Manage. Inf. Syst.* (5:4), 24:1-24:37. (<https://doi.org/10.1145/2629447>).
- Di Ciccio, C., Mecella, M., and Mendling, J. 2015. "The Effect of Noise on Mined Declarative Constraints," in *SIMPDA 2013*, pp. 1–24.
- Di Ciccio, C., Schouten, M. H. M., de Leoni, M., and Mendling, J. 2015. "Declarative Process Discovery with MINERful in ProM," in *CEUR Workshop Proceedings* (Vol. 1418), pp. 60–64. (<http://www.scopus.com/inward/record.url?eid=2-s2.0-84940398202&partnerID=40&md5=76831ac96819c6354c603636a08f723b>).
- Codasyl. 1982. "A Modern Appraisal Of Decision Tables," in *Report of the Decision Table Task Group*, ACM, New York, USA, pp. 230–232. (<https://doi.org/https://feb.kuleuven.be/prologa/CODASYL82/CODASYL82.pdf>).
- Curtis, B., Kellner, M. I., and Over, J. 1992. "Process Modeling," *Communications of the ACM - Special Issue on Analysis and Modeling in Software Development* (35:9), pp. 75–90. (<https://doi.org/10.1145/130994.130998>).
- Dadam, P., Reichert, M., and Kuhn, K. 2000. "Clinical Workflows - The Killer Application for Process-Oriented Information Systems?," in *Proc. of BIS 2000*, Poznan, Poland: Springer-Verlag, pp. 36–59. (<https://doi.org/10.1007/978-1-4471-0761-3>).
- Daelemans, W., Zavrel, J., van Der Sloot, K., and van Den Bosch, A. 2009. "TiMBL: Tilburg Memory Based Learner, Version 6.2, Reference Guide," *ILK Research Group Technical Report Series* (07–07), p. 66.
- Debois, S., Hildebrandt, T., and Slaats, T. 2015. "Concurrency and Asynchrony in Declarative Workflows," in *BPM'15. LNCS, Vol. 9253*, Innsbruck, Austria: Springer, pp. 72–89.
- Debois, S., Hildebrandt, T. T., Laursen, P. H., and Ulrik, K. R. 2017. *Declarative Process Mining for DCR Graphs*, pp. 759–764. (<https://doi.org/10.1145/3019612.3019622>).
- Dumas, M., van der Aalst, W. M. P., and ter Hofstede, A. H. M. 2005. *Process-Aware Information Systems: Bridging People and Software through Process Technology*, Wiley & Sons, New York, USA. (<https://books.google.be/books?hl=en&lr=&id=ZENNdQq8p74C&oi=fnd&pg=PR7&dq=dumas+process+aware+people+bridging&ots=ZhRO1VLRcG&sig=2jtp3Bn9Cd5BsSnlxnGDP-B2Aug#v=onepage&q=dumas+process+aware+people+bridging&f=false>).
- Dumas, M., La Rosa, M., Mendling, J., and Reijers, H. A. 2018. *Fundamentals of Business Process Management*, (2nd ed.), Springer-Verlag Berlin Heidelberg. (<https://doi.org/10.1007/978-3-662-56509-4>).
- van Eck, M. L., Lu, X., Leemans, S. J. J., and van der Aalst, W. M. P. 2015. "PM²: A Process Mining Project Methodology," in *CAiSE'15. LNCS, Vol. 9097*, pp. 297–313. (https://doi.org/10.1007/978-3-319-19069-3_19).
- Evermann, J., Rehse, J.-R., and Fettke, P. 2016. "A Deep Learning Approach for Predicting Process Behaviour at

- Runtime,” in *Proceedings of the 1st International Workshop on Runtime Analysis of Process-Aware Information Systems*, Springer.
- Faber, B., Konrad, R. A., Tang, C., and Trapp, A. C. 2016. “Examining the Impact of Regular Physician Visits on Heart Failure Patients: A Use Case with Electronic Health Data,” *Health Systems* (5:2), Nature Publishing Group, pp. 132–139. (<https://doi.org/10.1057/hs.2015.13>).
- Fahland, D., Lübke, D., Mendling, J., Reijers, H. a., Weber, B., Weidlich, M., and Zugal, S. 2009. “Declarative versus Imperative Process Modeling Languages: The Issue of Understandability,” in *BPMDS/EMMSAD’09 Workshop on Enterprise, Business-Process and Information Systems Modeling, Held at CAiSE’09. LNBIP, Vol. 29*, T. A. Halpin, J. Krogstie, S. Nurcan, E. Proper, R. Schmidt, P. Soffer, and R. Ukor (eds.), Amsterdam, The Netherlands: Springer, pp. 353–366. (http://link.springer.com/chapter/10.1007/978-3-642-01862-6_29).
- Fernandez-Llatas, C., Martinez-Millana, A., Martinez-Romero, A., Benedi, J. M., and Traver, V. 2015. “Diabetes Care Related Process Modelling Using Process Mining Techniques. Lessons Learned in the Application of Interactive Pattern Recognition: Coping with the Spaghetti Effect,” *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS* (2015-Novem), pp. 2127–2130. (<https://doi.org/10.1109/EMBC.2015.7318809>).
- Ferreira, H. M., and Ferreira, D. R. 2006. “An Integrated Life Cycle for Workflow Management Based on Learning and Planning,” *International Journal of Cooperative Information Systems* (15:04), pp. 485–505. (<https://doi.org/10.1142/S0218843006001463>).
- Ferri, C., Hernández-Orallo, J., and Modroi, R. 2009. “An Experimental Comparison of Performance Measures for Classification,” *Pattern Recognition Letters* (30:1), Elsevier B.V., pp. 27–38. (<https://doi.org/10.1016/j.patrec.2008.08.010>).
- Di Francescomarino, C., Dumas, M., Maggi, F. M., and Teinemaa, I. 2016. “Clustering-Based Predictive Process Monitoring,” *IEEE Transactions on Services Computing* (14:8). (<https://doi.org/10.1109/TSC.2016.2645153>).
- Frank, U. 2014. “Multi-Perspective Enterprise Modeling: Foundational Concepts, Prospects and Future Research Challenges,” *Software and Systems Modeling* (13:3), pp. 941–962. (<https://doi.org/10.1007/s10270-012-0273-9>).
- Funkner, A. A., Yakovlev, A. N., and Kovalchuk, S. V. 2017. “Towards Evolutionary Discovery of Typical Clinical Pathways in Electronic Health Records,” *Procedia Computer Science* (119:2017), Elsevier B.V., pp. 234–244. (<https://doi.org/10.1016/j.procs.2017.11.181>).
- Gawande, A. 2010. *The Checklist Manifesto*, Penguin Books India.
- Ghasemi, M., and Amyot, D. 2016. “Process Mining in Healthcare: A Systematised Literature Review,” *International Journal of Electronic Healthcare* (9:1), p. 60. (<https://doi.org/10.1504/IJEH.2016.078745>).
- Goedertier, S., Vanthienen, J., and Caron, F. 2015. “Declarative Business Process Modelling: Principles and Modelling Languages,” *Enterprise Information Systems* (9:2), pp. 161–185. (<https://doi.org/10.1080/17517575.2013.830340>).
- Greenes, R. A., Bates, D. W., Kawamoto, K., Middleton, B., Osherooff, J., and Shahar, Y. 2018. “Clinical Decision Support Models and Frameworks: Seeking to Address Research Issues Underlying Implementation Successes and Failures,” *Journal of Biomedical Informatics* (78:December 2017), pp. 134–143. (<https://doi.org/10.1016/j.jbi.2017.12.005>).
- Guha, S., Kettinger, W. J., and Teng, J. T. C. 1993. “Business Process Reengineering,” *Information Systems Management* (10:3), pp. 13–22. (<https://doi.org/10.1080/10580539308906939>).
- Guizzardi, G. 2013. “Ontology-Based Evaluation and Design of Visual Conceptual Modeling Languages,” in *Domain Engineering. Product Lines, Languages and Conceptual Models*, I. Reinhartz-Berger, A. Sturm, T. Clark, S. Cohen, and J. Bettin (eds.), Springer Berlin Heidelberg, pp. 317–347. (https://doi.org/10.1007/978-3-642-36654-3_13).
- Guizzardi, G., Wagner, G., De Almeida Falbo, R., Guizzardi, R. S. S., and Almeida, J. P. A. 2013. “Towards Ontological Foundations for the Conceptual Modeling of Events,” in *International Conference on Conceptual Modeling. LNCS, Vol. 8217*, pp. 327–341. (https://doi.org/10.1007/978-3-642-41924-9_27).
- Haarmann, S., Podlesny, N. J., Hewelt, M., Meyer, A., and Weske, M. 2015. “Production Case Management: A Prototypical Process Engine to Execute Flexible Business Processes,” in *Proceedings of the BPM Demo Session*, pp. 110–114.

- Haisjackl, C., Barba, I., Zugall, S., Soffer, P., Hadar, I., Reichert, M., Pinggera, J., and Weber, B. 2014. "Understanding Declare Models: Strategies, Pitfalls, Empirical Results," *Software & Systems Modeling* (Special Section Paper), pp. 1–28. (<https://doi.org/10.1007/s10270-014-0435-z>).
- Harris, Z. S. 1954. "Distributional Structure," *Word* (10:23), pp. 146–162. (<https://doi.org/10.1080/00437956.1954.11659520>).
- Hay, M. C., Weisner, T. S., Subramanian, S., Duan, N., Niedzinski, E. J., and Kravitz, R. L. 2008. "Harnessing Experience: Exploring the Gap between Evidence-Based Medicine and Clinical Practice," *Journal of Evaluation in Clinical Practice* (14:5), pp. 707–713. (<https://doi.org/10.1111/j.1365-2753.2008.01009.x>).
- Helfert, M. 2009. "Challenges of Business Processes Management in Healthcare: Experience in the Irish Healthcare Sector," *Business Process Management Journal* (15:6), pp. 937–952. (<https://doi.org/10.1108/14637150911003793>).
- Helm, E., and Paster, F. 2015. "First Steps Towards Process Mining in Distributed Health Information Systems," *International Journal of Electronics and Telecommunications* (61:2), pp. 137–142. (<https://doi.org/10.1515/eletel-2015-0017>).
- Hevner, A. R., March, S. T., Park, J., and Ram, S. 2004. "Design Science in Information Systems Research," *MIS Quarterly* (28:1), pp. 75–105.
- Hildebrandt, T., Mukkamala, R. R., and Slaats, T. 2012. "Declarative Modelling and Safe Distribution of Healthcare Workflows," in *FHIES 2011. LNCS, Vol. 7151*, Z. Liu and A. Wassung (eds.), Springer, Berlin, Heidelberg, pp. 39–56. (https://doi.org/10.1007/978-3-642-32355-3_3).
- Hofmann, P. B., and Perry, F. 2010. *Management Mistakes in Healthcare: Identification, Correction, and Prevention*, Cambridge University Press.
- Hollingsworth, D. 1995. "The Workflow Reference Model," *Workflow Management Coalition*. (<https://doi.org/citeulike-article-id:1378584>).
- Huang, Z., Dong, W., Bath, P., Ji, L., and Duan, H. 2015. "On Mining Latent Treatment Patterns from Electronic Medical Records," *Data Mining and Knowledge Discovery* (29:4), pp. 914–949. (<https://doi.org/10.1007/s10618-014-0381-y>).
- Huang, Z., Dong, W., Ji, L., Gan, C., Lu, X., and Duan, H. 2014. "Discovery of Clinical Pathway Patterns from Event Logs Using Probabilistic Topic Models," *Journal of Biomedical Informatics* (47), Elsevier Inc., pp. 39–57. (<https://doi.org/10.1016/j.jbi.2013.09.003>).
- Huang, Z., Dong, W., Ji, L., He, C., and Duan, H. 2016. "Incorporating Comorbidities into Latent Treatment Pattern Mining for Clinical Pathways," *Journal of Biomedical Informatics* (59), Elsevier Inc., pp. 227–239. (<https://doi.org/10.1016/j.jbi.2015.12.012>).
- Huang, Z., Lu, X., Duan, H., and Fan, W. 2013. "Summarizing Clinical Pathways from Event Logs," *Journal of Biomedical Informatics* (46:1), pp. 111–127. (<https://doi.org/10.1016/j.jbi.2012.10.001>).
- Hull, R., Damaggio, E., Fournier, F., Gupta, M., Heath, F., Hobson, S., Linehan, M. H., Maradugu, S., Nigam, A., Sukaviriya, P., and Vaculín, R. 2010. "Introducing the Guard-Stage-Milestone Approach for Specifying Business Entity Lifecycles," in *Web Services and Formal Methods. LNCS, Vol. 6551*, Springer, pp. 1–24. (https://doi.org/10.1007/978-3-642-19589-1_1).
- ISO/IEC. 2011. "ISO/IEC 9075-2:2011(En): Information Technology - Database Languages - SQL - Part 2: Foundation (SQL/Foundation)." (<https://www.iso.org/obp/ui/#iso:std:iso-iec:9075:-2:ed-4:v1:en>).
- Jensen, K. 1997. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use*, Berlin: Springer-Verlag.
- Jiménez-Ramírez, A., and Barba, I. 2013. "Generating Multi-Objective Optimized Business Process Enactment Plans," in *Advanced Information Systems Engineering. CAiSE'13. LNCS, Vol. 7908*, C. Salinesi, M. C. Norrie, and Ó. Pastor (eds.), Springer, pp. 99–115.
- Johnson, A. E. W., Pollard, T. J., Shen, L., Lehman, L. H., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Anthony Celi, L., and Mark, R. G. 2016. "MIMIC-III, a Freely Accessible Critical Care Database," *Scientific Data* (3), p. 160035. (<https://doi.org/10.1038/sdata.2016.35>).
- Keel, G., Savage, C., Rafiq, M., and Mazzocato, P. 2017. "Time-Driven Activity-Based Costing in Health Care: A Systematic Review of the Literature," *Health Policy* (121:7), Elsevier Ireland Ltd, pp. 755–763. (<https://doi.org/10.1016/j.healthpol.2017.04.013>).

- Klein, J. H., and Young, T. 2015. "Health Care: A Case of Hypercomplexity?," *Health Systems* (4:2), Springer, pp. 104–110.
- Kovalchuk, S., Funkner, A., Metsker, O., and Yakovlev, A. 2018. "Simulation of Patient Flow in Multiple Healthcare Units Using Process and Data Mining Techniques for Model Identification," *Journal of Biomedical Informatics* (82), Elsevier, pp. 128–142. (<https://doi.org/10.1016/j.jbi.2018.05.004>).
- Krogstie, J. 2000. "Evaluating UML: A Practical Application of a Framework for the Understanding of Quality in Requirements Specifications and Conceptual Modeling," in *Norwegian Informatics Conference (NIK)*.
- Krogstie, J. 2013. "Perspectives to Process Modeling," in *BPM. SCI, Vol. 444*, Springer, pp. 1–39.
- Kurniati, A. P., Johnson, O., Hogg, D., and Hall, G. 2016. "Process Mining in Oncology: A Literature Review," in *ICICM 2016*, IEEE, October, pp. 291–297. (<https://doi.org/10.1109/INFOCOMAN.2016.7784260>).
- Lanz, A., Posenato, R., Combi, C., and Reichert, M. 2014. "Controlling Time-Awareness in Modularized Processes," in *Proceedings of the 17th International Conference BPMDS 2016. LNBIP Vol. 175*, Springer, pp. 157–172. (https://doi.org/10.1007/978-3-319-39429-9_11).
- Lanz, A., Weber, B., and Reichert, M. 2014. "Time Patterns for Process-Aware Information Systems," *Requirements Engineering* (19:2), pp. 113–141. (<https://doi.org/10.1007/s00766-012-0162-3>).
- Lenz, R., Peleg, M., and Reichert, M. 2012. "Healthcare Process Support: Achievements, Challenges, Current Research," *International Journal of Knowledge-Based Organizations (IJKBO)* (4:2).
- Lenz, R., and Reichert, M. 2007. "IT Support for Healthcare Processes - Premises, Challenges, Perspectives," *Data and Knowledge Engineering* (61), pp. 39–58. (<https://doi.org/10.1016/j.datak.2006.04.007>).
- De Leoni, M., Van Der Aalst, W. M. P., and Dees, M. 2016. "A General Process Mining Framework for Correlating, Predicting and Clustering Dynamic Behavior Based on Event Logs," *Information Systems* (56), Elsevier, pp. 235–257. (<https://doi.org/10.1016/j.is.2015.07.003>).
- Leymann, F., and Roller, D. 2000. *Production Workflow - Concepts and Techniques*, Prentice Hall PTR.
- Lillrank, P., and Liukko, M. 2004. "Standard, Routine and Non-routine Processes in Health Care," *International Journal of Health Care Quality Assurance* (17:1), pp. 39–46. (<https://doi.org/10.1108/09526860410515927>).
- Lismont, J., Janssens, A. S., Odnoletkova, I., vanden Broucke, S., Caron, F., and Vanthienen, J. 2016. "A Guide for the Application of Analytics on Healthcare Processes: A Dynamic View on Patient Pathways," *Computers in Biology and Medicine* (77), Elsevier, pp. 125–134. (<https://doi.org/10.1016/j.combiomed.2016.08.007>).
- Lu, R., and Sadiq, S. W. 2007. "A Survey of Comparative Business Process Modeling Approaches," in *BIS'07. LNCS, Vol. 4439*, Poznan, Poland: Springer, pp. 82–94. (http://link.springer.com/chapter/10.1007/978-3-540-72035-5_7).
- Lu, R., Sadiq, S. W., and Governatori, G. 2009. "On Managing Business Processes Variants," *Data & Knowledge Engineering* (68:7), Elsevier B.V., pp. 642–664. (<https://doi.org/10.1016/j.datak.2009.02.009>).
- Madhavji, N. H., and Schafer, W. 1991. "Prism - Methodology and Process-Oriented Environment," *IEEE Transactions on Software Engineering* (17:12), pp. 1270–1283. (<https://doi.org/10.1109/32.106987>).
- Maggi, F. M. 2013. "Declarative Process Mining with the Declare Component of ProM," *CEUR Workshop Proceedings* (1021).
- Maggi, F. M. 2014. "Discovering Metric Temporal Business Constraints from Event Logs," *Lecture Notes in Business Information Processing* (194:August), pp. 261–275. (<https://doi.org/10.13140/2.1.4885.8888>).
- Maggi, F. M., Bose, R. P. J. C., and van der Aalst, W. M. P. 2013. "A Knowledge-Based Integrated Approach for Discovering and Repairing Declare Maps," in *CAiSE'13. LNCS, Vol. 7908*, Valencia, Spain, pp. 433–448. (https://doi.org/10.1007/978-3-642-38709-8_28).
- Maggi, F. M., Di Ciccio, C., Di Francescomarino, C., and Kala, T. 2018. "Parallel Algorithms for the Automated Discovery of Declarative Process Models," *Information Systems* (74), Elsevier Ltd, pp. 136–152. (<https://doi.org/10.1016/j.is.2017.12.002>).
- Maggi, F. M., Dumas, M., García-Bañuelos, L., and Montali, M. 2013. "Discovering Data-Aware Declarative Process Models from Event Logs," in *BPM'13. LNCS, Vol. 8094*, F. Daniel, J. Wang, and B. Weber (eds.), Springer, pp. 81–96. (https://doi.org/10.1007/978-3-642-40176-3_8).
- Maggi, F. M., Di Francescomarino, C., Dumas, M., and Ghidini, C. 2014. "Predictive Monitoring of Business Processes," *LNCS Vol. 8484*, pp. 457–472. (https://doi.org/10.1007/978-3-319-07881-6_31).

- Maggi, F. M., Jagadeesh Chandra Bose, R. P., and van der Aalst, W. M. P. 2012. "Efficient Discovery of Understandable Declarative Process Models from Event Logs," in *Advanced Information Systems Engineering. CAiSE'12. LNCS, Vol. 7328*, J. Ralyté, X. Franch, S. Brinkkemper, and S. Wrycza (eds.), Springer, pp. 270–285. (https://doi.org/10.1007/978-3-642-31095-9_18).
- Maggi, F. M., Montali, M., Di Ciccio, C., and Mendling, J. 2016. "Semantical Vacuity Detection in Declarative Process Mining," in *BPM'16. LNCS, Vol. 9850*, pp. 158–175. (https://doi.org/10.1007/978-3-319-45348-4_10).
- Maggi, F. M., Mooij, A. J., and van der Aalst, W. M. P. 2011. "User-Guided Discovery of Declarative Process Models," *CIDM 2011: IEEE Symposium on Computational Intelligence and Data Mining*, pp. 192–199. (<https://doi.org/10.1109/CIDM.2011.5949297>).
- Maggi, F. M., Slaats, T., and Reijers, H. A. 2014. "The Automated Discovery of Hybrid Processes," in *BPM'14. LNCS, Vol. 8659*, Haifa, Israel: Springer, pp. 392–399. (http://link.springer.com/chapter/10.1007/978-3-319-10172-9_27).
- Mans, R. S., Schonenberg, M. H., Song, M. S., van der Aalst, W. M. P., and Bakker, P. J. M. 2009. "Application of Process Mining in Healthcare - A Case Study in a Dutch Hospital," in *BIOSTEC 2009* (Vol. 25), Springer, pp. 425–438. (https://doi.org/10.1007/978-3-540-92219-3_32).
- Mariscal, G., Marbán, Ó., and Fernández, C. 2010. "A Survey of Data Mining and Knowledge Discovery Process Models and Methodologies," *Knowledge Engineering Review* (25:2), pp. 137–166. (<https://doi.org/10.1017/S0269888910000032>).
- Marquard, M., Shahzad, M., and Slaats, T. 2015. "Web-Based Modelling and Collaborative Simulation of Declarative Processes," in *BPM'15. LNCS, Vol. 9253*, Innsbruck, Austria: Springer, pp. 209–225. (https://doi.org/10.1007/978-3-319-23063-4_15).
- Mazzocato, P., Savage, C., Brommels, M., Aronsson, H., and Thor, J. 2010. "Lean Thinking in Healthcare: A Realist Review of the Literature," *BMJ Quality & Safety* (19:5), pp. 376–382. (<https://doi.org/10.1136/qshc.2009.037986>).
- Meier, J., Dietz, A., Boehm, A., and Neumuth, T. 2015. "Predicting Treatment Process Steps from Events," *Journal of Biomedical Informatics* (53), Elsevier Inc., pp. 308–319. (<https://doi.org/10.1016/j.jbi.2014.12.003>).
- Mertens, S., Gailly, F., and Poels, G. 2015. "Enhancing Declarative Process Models with DMN Decision Logic," in *Enterprise, Business-Process, and Information Systems Modelling at BPMDS/EMMSAD'15, Held at CAiSE'15. LNBIP, Vol. 214* (Vol. 214), Stockholm, Sweden: Springer, pp. 151–165. (https://doi.org/10.1007/978-3-319-19237-6_10).
- Mertens, S., Gailly, F., and Poels, G. 2017. "Towards a Decision-Aware Declarative Process Modeling Language for Knowledge-Intensive Processes," *Expert Systems with Applications* (87), Elsevier Ltd, pp. 316–334. (<https://doi.org/10.1016/j.eswa.2017.06.024>).
- Mertens, S., Gailly, F., and Poels, G. 2018. "Discovering Health-Care Processes Using DeciClareMiner," *Health Systems - Special Issue: SIG-Health Design, Innovation, and Impact of Healthcare IT* (7:3), Taylor & Francis, pp. 195–211. (<https://doi.org/10.1080/20476965.2017.1405876>).
- Mertens, S., Gailly, F., and Poels, G. 2019. "A Generic Framework for Flexible and Data-Aware Business Process Engines," in *FAISE Workshop at CAiSE'19. LNBIP, Vol. 349*, H. Proper and J. Stirna (eds.), Springer, Cham, pp. 201–213.
- Mertens, S., Gailly, F., Van Sassenbroeck, D., and Poels, G. 2019. "Comparing Strategies to Generate Experience-Based Clinical Process Recommendations That Leverage Similarity to Historic Data," in *2019 IEEE International Conference on Healthcare Informatics (ICHI)*, Xi'an, China: IEEE eXpress, pp. 73–83.
- Metzger, A., Leitner, P., Ivanovic, D., Schmieders, E., Franklin, R., Carro, M., Dustdar, S., and Pohl, K. 2015. "Comparing and Combining Predictive Business Process Monitoring Techniques," *IEEE Transactions on Systems, Man, and Cybernetics: Systems* (45:2), IEEE, pp. 276–290. (<https://doi.org/10.1109/TSMC.2014.2347265>).
- Mitchell, M. 1996. *Introduction to Genetic Algorithms*, MIT Press, Cambridge, MA.
- Montali, M. 2010. "Specification and Verification of Declarative Open Interaction Models," *A Logic-Based Approach. LNBIP, Vol. 56*, Springer. (<http://link.springer.com/content/pdf/10.1007/978-3-642-14538-4.pdf>).
- Montali, M., Chesani, F., Mello, P., and Maggi, F. M. 2013. "Towards Data-Aware Constraints in Declare," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing. SAC*, New York: ACM Press, pp.

- 1391–1396. (<https://doi.org/10.1145/2480362.2480624>).
- Moody, D. L. 2009. “The ‘Physics’ of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering,” *IEEE Transactions on Software Engineering* (35:6), pp. 756–779. (<http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5353439>).
- Mukkamala, R. R. 2012. “A Formal Model For Declarative Workflows - Dynamic Condition Response Graphs,” IT-Universitetet i København.
- Mulyar, N. 2005. “Pattern-Based Evaluation of Oracle-BPEL (v.10.1.2),” in *Technical Report BPM-05-24*, BPMcenter.Org.
- Mulyar, N., Pesic, M., van der Aalst, W. M. P., and Peleg, M. 2007. “Declarative and Procedural Approaches for Modelling Clinical Guidelines: Addressing Flexibility Issues,” in *BPM’07 Workshops. LNCS, Vol. 4928* (Vol. 4928), Brisbane, Australia: Springer, pp. 335–346. (http://link.springer.com/chapter/10.1007/978-3-540-78238-4_35).
- Mundbrod, N., and Reichert, M. 2017. “Configurable and Executable Task Structures Supporting Knowledge-Intensive Processes,” in *ER 2017: Conceptual Modeling. LNCS Vol. 10650*, pp. 388–402. (https://doi.org/10.1007/978-3-319-69904-2_30).
- Murata, T. 1989. “Petri Nets: Properties, Analysis and Applications,” *Proceedings of the IEEE*, pp. 541–580. (<https://doi.org/10.1109/5.24143>).
- Nelson, H. J., Poels, G., Genero, M., and Piattini, M. 2012. “A Conceptual Modeling Quality Framework,” *Software Quality Journal* (20:1), pp. 201–228. (<https://doi.org/10.1007/s11219-011-9136-9>).
- Netto, J. M., Barboza, T., Baião, F. A., and Santoro, F. M. 2019. “KiPN: A Visual Notation for Knowledge-Intensive Processes,” *International Journal of Business Process Integration and Management* (9:3).
- Noumeir, R. 2006. “Radiology Interpretation Process Modeling,” *Journal of Biomedical Informatics* (39:2), pp. 103–114. (<https://doi.org/10.1016/j.jbi.2005.07.001>).
- Object Management Group. 2013. “Business Process Model and Notation (BPMN2.02).” (<http://www.omg.org/spec/BPMN/>).
- Object Management Group. 2014. “Decision Model and Notation (DMN1.0).” (<http://www.omg.org/spec/DMN/Current/>).
- Object Management Group. 2015. “Unified Modeling Language (UML).” (<http://www.omg.org/spec/UML/>).
- Object Management Group. 2016. “Decision Model and Notation (DMN1.1),” (Vol. 1.0). (<http://www.omg.org/spec/DMN/>).
- Orellana Garcia, A., Perez Ramirez, Y. E., and Armenteros Larrea, O. U. 2015. “Process Mining in Healthcare: Analysis and Modeling of Processes in the Emergency Area,” *IEEE Latin America Transactions* (13:5), pp. 1612–1618. (<https://doi.org/10.1109/TLA.2015.7112022>).
- Payton, F. C., Paré, G., LeRouge, C., and Reddy, M. 2011. “Health Care IT: Process, People, Patients and Interdisciplinary Considerations,” *Journal of the Association for Information Systems* (12:2).
- Peffer, K., Rothenberger, M. A., Tuunanen, T., and Vaezi, R. 2012. “Design Science Research Evaluation,” *Design Science Research in Information Systems. Advances in Theory and Practice*, pp. 398–410. (https://doi.org/10.1007/978-3-642-29863-9_29).
- Peffer, K., Tuunanen, T., Rothenberger, M. A., and Chatterjee, S. 2007. “A Design Science Research Methodology for Information Systems Research,” *Journal of Management Information Systems* (24:3), M.E. Sharpe Inc., pp. 45–77.
- Perer, A., Wang, F., and Hu, J. 2015. “Mining and Exploring Care Pathways from Electronic Medical Records with Visual Analytics,” *Journal of Biomedical Informatics* (56), Elsevier Inc., pp. 369–378. (<https://doi.org/10.1016/j.jbi.2015.06.020>).
- Pesic, M. 2008. “Constraint-Based Workflow Management Systems: Shifting Control to Users,” Technische Universiteit Eindhoven. (<https://doi.org/http://www.narcis.nl/publication/RecordID/oai:library.tue.nl:638413>).
- Petri, C. A. 1962. “Kommunikation Mit Automaten,” Universität Bonn.
- Pryss, R., Mundbrod, N., Langer, D., and Reichert, M. 2015. “Supporting Medical Ward Rounds through Mobile Task and Process Management,” in *Information Systems and E-Business Management* (Vol. 13), pp. 107–146. (<https://doi.org/10.1007/s10257-014-0244-5>).

- Putman, K., Anstey, A., Harper, P. R., and Knight, V. a. 2015. "Modelling of Psoriasis Patient Flows for the Reconfiguration of Secondary Care Services and Treatments," *Health Systems* (January), pp. 1–8. (<https://doi.org/10.1057/hs.2015.4>).
- Rashid, M., Naeem, H., Aamir, M., Ali, W., and Ahmed, W. 2017. "A Multi-Level Process Mining Framework for Correlating and Clustering of Biomedical Activities Using Event Logs," *International Journal of Advanced Computer Science and Applications* (8:3). (<https://doi.org/10.14569/IJACSA.2017.080354>).
- Rebuge, Á., and Ferreira, D. R. 2012. "Business Process Analysis in Healthcare Environments: A Methodology Based on Process Mining," *Information Systems* (37:2), pp. 99–116. (<https://doi.org/10.1016/j.is.2011.01.003>).
- Recker, J., Rosemann, M., Indulska, M., and Green, P. 2009. "Business Process Modeling- A Comparative Analysis," *Journal of the Association for Information Systems* (10:4), pp. 333–363. (<http://aisel.aisnet.org/cgi/viewcontent.cgi?article=1501&context=jais>).
- Reichert, M., and Weber, B. 2012. *Enabling Flexibility in Process-Aware Information Systems: Challenges, Methods, Technologies*, Springer.
- Rismanchian, F., and Lee, Y. H. 2017. "Process Mining–Based Method of Designing and Optimizing the Layouts of Emergency Departments in Hospitals," *Health Environments Research and Design Journal* (10:4), pp. 105–120. (<https://doi.org/10.1177/1937586716674471>).
- Rojas, E., Arias, M., and Sepúlveda, M. 2015. "Clinical Processes and Its Data, What Can We Do with Them?," in *HEALTHINF '15*, SCITEPRESS - Science and Technology Publications, pp. 642–647. (<https://doi.org/10.5220/0005287206420647>).
- Rojas, E., Munoz-Gama, J., Sepúlveda, M., and Capurro, D. 2016. "Process Mining in Healthcare: A Literature Review," *Journal of Biomedical Informatics* (61), Elsevier Inc., pp. 224–236. (<https://doi.org/10.1016/j.jbi.2016.04.007>).
- Rojo, M. G., Rolón, E., Calahorra, L., García, F. O., Sánchez, R. P., Ruiz, F., Ballester, N., Armenteros, M., Rodríguez, T., and Espartero, R. M. 2008. "Implementation of the Business Process Modelling Notation (BPMN) in the Modelling of Anatomic Pathology Processes.," *Diagnostic Pathology* (3 Suppl 1), p. S22. (<https://doi.org/10.1186/1746-1596-3-S1-S22>).
- Rotter, T., Kinsman, L., James, E. L., Machotta, A., Gothe, H., Willis, J., Snow, P., and Kugler, J. 2010. "Clinical Pathways: Effects on Professional Practice, Patient Outcomes, Length of Stay and Hospital Costs," *Cochrane Database of Systematic Reviews* (3), (T. Rotter, ed.), Chichester, UK: John Wiley & Sons, Ltd, pp. 1–170. (<https://doi.org/10.1002/14651858.CD006632.pub2>).
- Rovani, M., Maggi, F. M., de Leoni, M., and van der Aalst, W. M. P. 2015. "Declarative Process Mining in Healthcare," *Expert Systems with Applications* (42:23), Elsevier Ltd., pp. 9236–9251. (<https://doi.org/10.1016/j.eswa.2015.07.040>).
- Rowley, J. 2007. "The Wisdom Hierarchy: Representations of the DIKW Hierarchy," *Journal of Information Science* (33:2), pp. 163–180. (<https://doi.org/10.1177/0165551506070706>).
- Russell, N., van der Aalst, W. M. P., ter Hofstede, A. H. M., and Edmond, D. 2005. "Workflow Resource Patterns: Identification, Representation and Tool Support," in *Advanced Information Systems Engineering. LNCS, Vol. 3520*, Springer, pp. 216–232. (https://doi.org/10.1007/11431855_16).
- Russell, N., ter Hofstede, A. H. M., Edmond, D., and van der Aalst, W. M. P. 2004. "Workflow Resource Patterns," *BETA Working Paper Series, WP 127*.
- Sadiq, S. W., Orłowska, M. E., and Sadiq, W. 2005. "Specification and Validation of Process Constraints for Flexible Workflows," *Information Systems* (30:5), pp. 349–378. (<https://doi.org/10.1016/j.is.2004.05.002>).
- Schonenberg, H., Mans, R., Russell, N., Mulyar, N., and van der Aalst, W. M. P. 2008. "Process Flexibility: A Survey of Contemporary Approaches," in *CIAO! 2008, EOMAS 2008. LNBIP, Vol. 10*, Springer, pp. 16–30. (https://doi.org/10.1007/978-3-540-68644-6_2).
- Schonenberg, H., and Weber, B. 2008. "Supporting Flexible Processes through Recommendations Based on History," in *BPM'08. LNCS, Vol. 5240* (Vol. 5240), Milan, Italy: Springer, pp. 51–66. (http://link.springer.com/chapter/10.1007/978-3-540-85758-7_7).
- Schönig, S., Ciccio, C. Di, Maggi, F. M., and Mendling, J. 2016. "Discovery of Multi-Perspective Declarative Process Models," in *ICSOC 2016: Service-Oriented Computing. LNCS, Vol. 9936*, Q. Sheng, E. Stroulia, S. Tata, and S. Bhiri (eds.), Springer. (https://doi.org/https://doi.org/10.1007/978-3-319-46295-0_6).

- Sein, M. K., Henfridsson, O., Purao, S., Rossi, M., and Lindgren, R. 2011. "Action Design Research," *MIS Quarterly* (35:1), p. 37. (<https://doi.org/10.2307/23043488>).
- Shearer, C. 2000. "The Crisp-Dm Model: The New Blueprint for Data Mining," *Journal of Data Warehousing* (5:4), pp. 13–22.
- Shostack, G. L. 1987. "Positioning Through Structural Change," *Journal of Marketing* (51:1), pp. 34–43.
- Sittig, D. F., Wright, A., Osheroff, J. A., Middleton, B., Teich, J. M., Ash, J. S., Campbell, E., and Bates, D. W. 2008. "Grand Challenges in Clinical Decision Support," *Journal of Biomedical Informatics* (41:2), pp. 387–392. (<https://doi.org/10.1016/j.jbi.2007.09.003>).
- Slaats, T., Mukkamala, R. R., Hildebrandt, T., and Marquard, M. 2013. "Exformatics Declarative Case Management Workflows as DCR Graphs," in *BPM'13. LNCS, Vol. 8094* (Vol. 8094 LNCS), Beijing, China: Springer, pp. 339–354. (https://doi.org/10.1007/978-3-642-40176-3_28).
- De Smedt, J., De Weerd, J., and Vanthienen, J. 2015. "Fusion Miner: Process Discovery for Mixed-Paradigm Models," *Decision Support Systems* (77), Elsevier B.V., pp. 123–136. (<https://doi.org/10.1016/j.dss.2015.06.002>).
- De Smedt, J., De Weerd, J., Vanthienen, J., and Poels, G. 2016. "Mixed-Paradigm Process Modeling with Intertwined State Spaces," *Business & Information Systems Engineering* (58:1), Springer Fachmedien Wiesbaden, pp. 19–29. (<https://doi.org/10.1007/s12599-015-0416-y>).
- Song, M., Günther, C. W., and van der Aalst, W. M. P. 2009. "Trace Clustering in Process Mining," in *BPM'08. LNBIP, Vol. 17* (Vol. 17), D. Ardagna, M. Mecella, and J. Yang (eds.), pp. 109–120. (https://doi.org/10.1007/978-3-642-00328-8_11).
- Song, M., Yang, H., Siadat, S. H., and Pechenizkiy, M. 2013. "A Comparative Study of Dimensionality Reduction Techniques to Enhance Trace Clustering Performances," *Expert Systems with Applications* (40:9), pp. 3722–3737. (<https://doi.org/10.1016/j.eswa.2012.12.078>).
- van der Spoel, S., van Keulen, M., and Amrit, C. 2013. "Process Prediction in Noisy Data Sets: A Case Study in a Dutch Hospital," in *SIMPDA'12. LNBIP Vol. 162*, pp. 60–83. (https://doi.org/10.1007/978-3-642-40919-6_4).
- Stefanelli, M. 2004. "Knowledge and Process Management in Health Care Organizations," *Methods of Information in Medicine* (43:5), pp. 525–535. (<http://www.embase.com/search/results?subaction=viewrecord&from=export&id=L40040337>).
- Strnadl, C. F. . 2006. "Aligning Business and IT: The Process-Driven Architecture Model," *Information Systems Management* (23:4), pp. 67–77. (<http://w3.bgu.ac.il/lib/customproxy.php?url=http://search.ebscohost.com/login.aspx?direct=true&db=lls&AN=22291669&site=eds-live&authtype=ip,uid&custid=s4309548&groupid=main&profile=eds>).
- Suriadi, S., Andrews, R., ter Hofstede, A. H. M., and Wynn, M. T. 2017. "Event Log Imperfection Patterns for Process Mining: Towards a Systematic Approach to Cleaning Event Logs," *Information Systems* (64:October 2015), Elsevier, pp. 132–150. (<https://doi.org/10.1016/j.is.2016.07.011>).
- Svagård, I., and Farshchian, B. A. 2009. "Using Business Process Modelling to Model Integrated Care Processes: Experiences from a European Project," in *IWANN 2009* (Vol. pt.2), pp. 922–925. (http://dx.doi.org/10.1007/978-3-642-02481-8_140).
- Tax, N., Teinemaa, I., and Zelst, S. J. Van. (n.d.). "An Interdisciplinary Comparison of Sequence Modeling Methods for Next-Element Prediction," *ArXiv Machine Learning*. (<https://doi.org/arXiv:1811.00062v1>).
- Tax, N., Verenich, I., Rosa, M. La, and Dumas, M. 2017. "Predictive Business Process Monitoring with LSTM Neural Networks," in *CAiSE 2017. LNCS Vol. 10253*, E. Dubois and K. Pohl (eds.), Springer, pp. 477–492. (<https://doi.org/10.1007/978-3-319-59536-8>).
- Taylor, J., Fish, F. A., and Vanthienen, J. 2013. "Emerging Standards in Decision Modeling - an Introduction to Decision Model & Notation," in *IBPMS: Intelligent BPM Systems: Intelligent BPM Systems: Impact and Opportunity*, pp. 133–146. (<https://lirias.kuleuven.be/bitstream/123456789/424966/1/14+Emerging+Standards+in+Decision+Modeling-Taylor+et+al+final.pdf>).
- Trkman, P. 2010. "The Critical Success Factors of Business Process Management," *International Journal of Information Management* (30:2), pp. 125–134. (<https://doi.org/10.1016/j.ijinfomgt.2009.07.003>).
- Vanhaecht, K., De Witte, K., Depreitere, R., and Sermeus, W. 2006. "Clinical Pathway Audit Tools: A Systematic

- Review,” *Journal of Nursing Management* (14:7), pp. 529–537. (<https://doi.org/10.1111/j.1365-2934.2006.00705.x>).
- Vanthienen, J. 2014. “Introducing a Proposal to Standardize a Decision Model and Notation.” (http://www.omg.org/news/meetings/tc/agendas/va/DMN_pdf/Vincent_SainteMarie_vanThienen.pdf).
- Vassilakopoulou, P., Ponis, S. T., Gayialis, S. P., Papadopoulos, G. A., and Tatsiopoulos, I. P. 2015. “Integrating Process Modeling and Simulation with Benchmarking Using a Business Process Management System for Local Government,” *International Journal of Computer Theory and Engineering* (8:6), pp. 482–489. (<https://doi.org/10.7763/ijcte.2016.v8.1093>).
- Wand, Y., Storey, V. C., and Weber, R. 1999. “An Ontological Analysis of the Relationship Construct in Conceptual Modeling,” *ACM Trans. Database Syst.* (24:4), pp. 494–528. (<https://doi.org/10.1145/331983.331989>).
- Wang, T., Tian, X., Yu, M., Qi, X., and Yang, L. 2017. “Stage Division and Pattern Discovery of Complex Patient Care Processes,” *Journal of Systems Science and Complexity* (30:5), pp. 1136–1159. (<https://doi.org/10.1007/s11424-017-5302-x>).
- Wastell, D. G., White, P., and Kawalek, P. 1994. “A Methodology for Business Process Redesign: Experiences and Issues,” *The Journal of Strategic Information Systems* (3:1), pp. 23–40. ([https://doi.org/10.1016/0963-8687\(94\)90004-3](https://doi.org/10.1016/0963-8687(94)90004-3)).
- Weske, M. 2012. *Business Process Management: Concepts, Languages, Architectures*, Springer. (<https://doi.org/10.1007/978-3-642-28616-2>).
- Westergaard, M., and Maggi, F. M. 2012. “Looking Into the Future about Timed Declarative Process Models,” in *OTM 2012* (Vol. 7565), Springer, pp. 250–267. (https://doi.org/10.1007/978-3-642-33606-5_16).
- White, S. A. 2004. “Introduction to BPMN,” *IBM Cooperation*, pp. 1–11. (http://yoann.nogues.free.fr/IMG/pdf/07-04_WP_Intro_to_BPMN_-_White-2.pdf).
- Wieringa, R. J. 2009. “Design Science as Nested Problem Solving,” in *Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology - DESRIST '09*, New York: ACM Press, pp. 1–12. (<https://doi.org/10.1145/1555619.1555630>).
- Wieringa, R. J., and Heerkens, J. M. G. 2006. “The Methodological Soundness of Requirements Engineering Papers: A Conceptual Framework and Two Case Studies,” *Requirements Engineering* (11:4), pp. 295–307. (<https://doi.org/10.1007/s00766-006-0037-6>).
- Wohed, P., Russell, N., ter Hofstede, A. H. M., Andersson, B., and van der Aalst, W. M. P. 2009. “Patterns-Based Evaluation of Open Source BPM Systems: The Cases of JBPM, OpenWFE, and Enhydra Shark,” *Information and Software Technology* (51:8), pp. 1187–1216. (<https://doi.org/10.1016/j.infsof.2009.02.002>).
- Wyatt, J. C. 2001. “Management of Explicit and Tacit Knowledge,” *Journal of the Royal Society of Medicine* (94:1), pp. 6–9. (<https://doi.org/10.1177/014107680109400102>).

Appendix A: Arm fracture case description (based on literature)

The process of treating arm-related fractures takes place in the emergency and orthopedic department of a hospital. The process entails the registration, diagnosis and treatment phases for patients with suspected fractures of a finger, hand, wrist, forearm, upper arm, shoulder, and/or collarbone.

The process starts when a patient is registered at the reception of the emergency department. Alternatively, in acute emergency situations, the registration can be done at a later time. The next step will usually be to take a medical history (e.g., previous injuries and medication) and perform a clinical examination of the patient. During this the doctor will make enquiries as to the current symptoms (e.g., excessive pain and deformation) of the patient and perform a hands-on examination. Based on these symptoms the doctor will make a preliminary diagnosis. Normally, this diagnosis is checked by making X-rays, which in turn always results in a new doctor-patient consultation to evaluate the X-rays and make the actual diagnosis. In some situations, the doctor can make the final diagnosis without X-rays (e.g., clearly no fracture). In some cases, a complementary CT-scan will be used to get even more information.

The next phase involves the treatment the patient will receive. Of course, a treatment is only possible after a preliminary clinical examination by a doctor. There are 9 possible treatments: prescribing rest (=no treatment), applying ice, managing pain with pain medication, applying a bandage, applying a splint, providing support with a sling, fixating the fracture, applying a cast or performing surgery. Choosing one does not eliminate other treatments, as some treatment strategies combine two or more treatments. It is also possible that ice, a sling, some bandages and/or a splint is applied at any time during the process in order to make the patient as comfortable as possible, no matter the diagnosis.

When we look at the case on a more detailed level we can identify several different variations. These represent the classes of fractures that can occur. Each has a specific flow and different characteristics to be taken into account. One common characteristic is that all fractures require surgery if the fracture is open or displaced. Also, if there is no emergency situation, a fracture diagnosis will need to be confirmed by an X-ray (usually also in emergency situations depending on the type of injury).

- A fractured finger or a fractured bone in his hand: in most cases a simple fixation is enough to let it heal. The patient will receive a sling before being sent home.
- A fractured wrist: a cast will be applied, possibly after performing surgery. Surgery is required if the patient is a child (less than 15 years old) and has a damaged periosteum. For adults, surgery is only performed when dealing with open or complex fractures. Afterwards a follow-up X-ray will be taken to confirm that the bone is positioned correctly to start the healing process. The patient will receive a sling before being sent home.

- A fractured forearm: usually this requires no more than a cast. Only when the bone parts are too far apart, surgery is required. To support the cast, the patient receives a sling before being sent home.
- A fractured upper arm: is commonly treated by applying a fixation. The patient will receive a sling before being sent home.
- A fractured shoulder: usually the conservative treatment is enough, letting the shoulder heal while wearing a sling. In the other cases, surgery is required. Physio-therapy is also needed, because the shoulder joint will be inactive for an extended period during either of both treatments.
- A fractured collarbone: is treated in most cases by resting it while wearing a figure of eight bandage. Surgery is only required when dealing with open or complex fractures or extensive damage to the arteries or nerves.

Additionally, if surgery is required for a broken wrist or forearm, but the OR is unavailable, a temporary cast will be applied to bridge the time until surgery can be performed.

Another aspect is the prescription of medication. There is a general policy that states that no medication can be prescribed without being preceded by an actual doctor's examination. For pain medication it also requires the doctor or surgeon to agree that the patient is in pain or could be in pain in the nearby future. After surgery patients will be prescribed anticoagulants and a painkiller as precaution. In many cases an additional stomach protecting drug is given to counteract possible detrimental side effects of the other two prescriptions. Furthermore, patients that received a cast could be prescribed anticoagulants. Because there exist strong painkillers that do not mix well with anticoagulants, a distinction is made between classes of painkillers:

- Painkillers A: should not be taken while on anticoagulants but are preferred in other cases.
- Painkillers B: can be taken while on anticoagulants.

Additionally, we also need to consider that some activities require the availability of certain resources, which in turn are limited in number. Also, they are not only used for this process, but rather represent a pool of resources shared among multiple independent processes of the hospital. The inventory of the material resources as used during the process goes as follows:

- Reception desks
- Exam rooms
- X-ray rooms with an X-ray machine
- CT-scan room with an CT-scanner

- Operating rooms
- Recovery room beds
- Patient rooms
- Patient beds
- Physiotherapy room

Furthermore, there is also a list of human resources available:

- Receptionists
- Nurses
- Doctors
 - Anesthesiologist
 - Orthopedic surgeon
 - Emergency doctors
- Surgical assistant
- Physiotherapist

Some of the assumptions made:

- Patients do not go home until the process instance is finished
- Other medical issues are out of scope
- Patients do not refuse the recommended treatments

Appendix B: Original textual DeciClare model of arm fracture case

Available resources

[nurse, doctor, receptionist, surgeon, physiotherapist, reception desk, examination room, x-ray room, operating room, patient bed, physiotherapy room]

Available activities

[Register patient, Examine patient, Take X-ray, Prescribe weak painkillers, Prescribe strong painkillers A, Prescribe strong painkillers B, Prescribe anticoagulants, Prescribe anti-inflammatory drugs, Apply cast, Apply sling, Apply fixation, Apply bandage, Apply figure of eight bandage, Perform surgery, Let patient rest, Perform physiotherapy]

Available data elements

[Clear emergency?=(True/False), Where there complications during surgery?=(True/False), Open or complex fracture?=(True/False), Extensive damage to arteries or nerves?=(True/False), Both arms broken?=(True/False), No improvement in 3 months?=(True/False), More than 16yrs old?=(True/False), Periosteum torn?=(True/False), OR available?=(True/False), Bone parts far apart?=(True/False), In pain (now or in foreseeable future)?=(True/False), On weak painkillers?=(True/False), On anticoagulants?=(True/False), On anti-inflammatory drugs?=(True/False), Fractured bone?=(Wrist/Forearm/Shoulder/Finger/Upper arm/Hand/Collarbone)]

Resource constraints

- there are 10 nurses available
- there are 3 doctors available
- there are 2 receptionists available
- there are 2 surgeons available
- there is 1 physiotherapist available
- there are 2 reception desks available
- there are 15 examination rooms available
- there are 3 operating rooms available
- there is 1 x-ray room available
- there are 60 patient beds available
- there is 1 physiotherapy room available
- the execution of activity 'Register patient' requires a 'reception desk'
- the execution of activity 'Register patient' requires a 'receptionist'
- *If* [*'Clear emergency?' = false*] *then*
the execution of activity 'Examine patient' requires an 'examination room'
- the execution of activity 'Examine patient' requires a 'doctor'
- the execution of activity 'Take X-ray' requires a 'x-ray room'
- the execution of activity 'Take X-ray' requires a 'doctor' or 'nurse'
- the execution of activity 'Apply cast' requires an 'examination room'
- the execution of activity 'Apply cast' requires a 'doctor' or 'nurse'
- the execution of activity 'Apply fixation' requires an 'examination room'
- the execution of activity 'Apply fixation' requires a 'doctor' or 'nurse'
- the execution of activity 'Apply figure of eight bandage' requires an 'examination room'
- the execution of activity 'Apply figure of eight bandage' requires a 'doctor' or 'nurse'
- the execution of activity 'Apply bandage' requires an 'examination room'
- the execution of activity 'Apply bandage' requires a 'doctor' or 'nurse'
- the execution of activity 'Apply sling' requires an 'examination room'
- the execution of activity 'Apply sling' requires a 'doctor' or 'nurse'
- the execution of activity 'Perform surgery' requires an 'operating room'
- the execution of activity 'Perform surgery' requires a 'surgeon'
- the execution of activity 'Let patient rest' requires a 'patient bed'

- the execution of activity 'Perform physiotherapy' requires a 'physiotherapy room'
- the execution of activity 'Perform physiotherapy' requires a 'physiotherapist'
- the execution of activity 'Prescribe weak painkillers' requires a 'doctor'
- the execution of activity 'Prescribe strong painkillers A' requires a 'doctor'
- the execution of activity 'Prescribe strong painkillers B' requires a 'doctor'
- the execution of activity 'Prescribe anticoagulants' requires a 'doctor'
- the execution of activity 'Prescribe anti-inflammatory drugs' requires a 'doctor'

Control-flow constraints

- *If[['Clear emergency? = false']] then*
'Register patient' has to be performed as the first activity
- 'Register patient' has to be performed at least 1 time
- 'Register patient' can be performed at most 1 time
- 'Examine patient' has to be performed at least 1 time
- 'Apply sling' can be performed at most 2 times
- at least 1 out of 6 of the following activities have to be performed: 'Apply sling', 'Apply figure of eight bandage', 'Apply fixation', 'Apply cast', 'Perform surgery' and 'Apply bandage'
- *If[['Clear emergency? = false']] then*
'Register patient' has to precede 'Examine patient'
- 'Examine patient' has to precede 'Take X-ray'
- 'Take X-ray' has to be followed by 'Examine patient'
- 'Examine patient' has to precede 'Prescribe weak painkillers'
- 'Examine patient' has to precede 'Prescribe strong painkillers A'
- 'Examine patient' has to precede 'Prescribe strong painkillers B'
- 'Examine patient' has to precede 'Prescribe anticoagulants'
- 'Examine patient' has to precede 'Perform surgery'
- 'Examine patient' has to precede 'Apply fixation'
- 'Examine patient' has to precede 'Apply cast'
- 'Examine patient' has to precede 'Apply figure of eight bandage'
- 'Prescribe strong painkillers A' cannot be followed by 'Prescribe anticoagulants' within at most 5 days
- 'Prescribe anticoagulants' cannot be followed by 'Prescribe strong painkillers A' within at most 5 days
- *If[['Where there complications during surgery? = false']] then*
'Perform surgery' has to be followed by 'Let patient rest' within at most 5 minutes
- *If[['Open or complex fracture? = false' and 'Fractured bone? = Upper arm' and 'Extensive damage to arteries or nerves? = false' and 'Both arms broken? = false' and 'No improvement in 3 months? = false'] or ['Open or complex fracture? = false' and 'Fractured bone? = Hand' and 'Extensive damage to arteries or nerves? = false'] or ['Open or complex fracture? = false' and 'Fractured bone? = Finger' and 'Extensive damage to arteries or nerves? = false']] then*
'Apply fixation' has to be performed at least 1 time
- *If[['Open or complex fracture? = false' and 'Fractured bone? = Forearm' and 'Extensive damage to arteries or nerves? = false'] or ['Open or complex fracture? = false' and 'Extensive damage to arteries or nerves? = false' and 'More than 16yrs old? = false' and 'Fractured bone? = Wrist' and 'Periosteum torn? = false'] or ['Open or complex fracture? = false' and 'Extensive damage to arteries or nerves? = false' and 'Fractured bone? = Wrist' and 'More than 16yrs old? = true'] or ['Open or complex fracture? = false' and 'Fractured bone? = Shoulder' and 'Extensive damage to arteries or nerves? = false']] then*
'Apply cast' has to be performed at least 1 time
- *If[['Extensive damage to arteries or nerves? = true' and 'Fractured bone? = Forearm' and 'OR available? = false'] or ['Extensive damage to arteries or nerves? = true' and 'Fractured bone? = Wrist' and 'OR available? = false'] or ['Open or complex fracture? = true' and 'Fractured bone? = Wrist' and 'OR available? = false'] or ['Fractured bone? = Forearm' and 'Open or complex fracture? = true' and 'OR available? = false']] then*
'Apply cast' has to be performed at least 2 times

- *If[['Open or complex fracture? = false' and 'Extensive damage to arteries or nerves? = false' and 'Fractured bone? = Collarbone']] then*
'Apply figure of eight bandage' has to be performed at least 1 time
- *If[['Open or complex fracture? = false' and 'Periosteum torn? = true' and 'Extensive damage to arteries or nerves? = false' and 'More than 16yrs old? = false' and 'Fractured bone? = Wrist'] or ['Extensive damage to arteries or nerves? = true' and 'Fractured bone? = Upper arm' and 'Extensive damage to arteries or nerves? = false'] or ['Extensive damage to arteries or nerves? = true' and 'Fractured bone? = Forearm' and 'OR available? = false'] or ['Open or complex fracture? = true'] or ['Extensive damage to arteries or nerves? = true' and 'Fractured bone? = Wrist' and 'OR available? = false'] or ['Open or complex fracture? = false' and 'Fractured bone? = Upper arm' and 'Extensive damage to arteries or nerves? = false' and 'Both arms broken? = true']] then*
'Perform surgery' has to be performed at least 1 time
- *If[['Fractured bone? = Forearm' and 'Open or complex fracture? = true'] or ['Open or complex fracture? = true' and 'Fractured bone? = Wrist'] or ['Extensive damage to arteries or nerves? = true' and 'Fractured bone? = Forearm'] or ['Extensive damage to arteries or nerves? = true' and 'Fractured bone? = Wrist']] then*
'Perform surgery' has to be followed by 'Apply cast'
- *If[['Fractured bone? = Hand'] or ['Fractured bone? = Upper arm'] or ['Fractured bone? = Finger'] or ['Fractured bone? = Shoulder'] or ['Fractured bone? = Forearm'] or ['Fractured bone? = Wrist']] then*
'Apply sling' has to be performed at least 1 time
- *If[['In pain (now or in foreseeable future)? = true' and 'On anti-inflammatory drugs? = false' and 'On anticoagulants? = false' and 'On weak painkillers? = true']] then*
'Perform surgery' or 'Examine patient' have to be followed by 'Prescribe strong painkillers A'
- *If[['On anti-inflammatory drugs? = true' and 'In pain (now or in foreseeable future)? = true' and 'More than 16yrs old? = false' and 'On weak painkillers? = true'] or ['In pain (now or in foreseeable future)? = true' and 'More than 16yrs old? = true' and 'On anticoagulants? = true'] or ['On anti-inflammatory drugs? = true' and 'In pain (now or in foreseeable future)? = true' and 'More than 16yrs old? = true'] or ['In pain (now or in foreseeable future)? = true' and 'More than 16yrs old? = false' and 'On weak painkillers? = true' and 'On anticoagulants? = true']] then*
'Perform surgery' or 'Examine patient' have to be followed by 'Prescribe strong painkillers B'
- *If[['On weak painkillers? = false' and 'In pain (now or in foreseeable future)? = true' and 'More than 16yrs old? = false']] then*
'Perform surgery' or 'Examine patient' have to be followed by 'Prescribe weak painkillers'
- 'Perform surgery' has to be followed by 'Prescribe anticoagulants' within at most 4 hours
- 'Perform surgery' has to be followed by 'Prescribe weak painkillers' or 'Prescribe strong painkillers B' within at most 4 hours
- *If[['Fractured bone? = Shoulder' and 'Clear emergency? = false'] or ['Clear emergency? = false' and 'Fractured bone? = Hand'] or ['Clear emergency? = false' and 'Fractured bone? = Wrist'] or ['Clear emergency? = false' and 'Fractured bone? = Forearm'] or ['Clear emergency? = false' and 'Fractured bone? = Collarbone'] or ['Clear emergency? = false' and 'Fractured bone? = Finger'] or ['Clear emergency? = false' and 'Fractured bone? = Upper arm']] then*
'Take X-ray' has to precede 'Apply figure of eight bandage' or 'Apply fixation' or 'Apply cast' or 'Perform surgery'
- 'Apply cast' has to be followed by 'Perform surgery' or 'Take X-ray'
- *If[['Fractured bone? = Shoulder']] then*
'Perform surgery' has to be followed by 'Perform physiotherapy'

Appendix C: Semi-structured interview protocol (in Dutch)

Semi-structured interview:

A) Introductie

1. Introductie van mezelf
2. Beschrijving van het onderzoek
 - a. Kennisintensieve, mens-centrische en flexibele processen (zoals veel processen in de gezondheidszorg) zijn processen waarbij set van activiteiten hebt die op vele verschillende manieren uitgevoerd kunnen worden (=variaties). Voorbeeld...
 - b. Dokters en verpleegkundige voeren het proces uit en beslissen welke activiteiten uit te voeren en wanneer (=leiden)
 - c. Om die beslissingen te maken gebruiken deze wat men tacit knowledge: een impliciet idee van hoe het proces moet verlopen in elk specifiek geval (=condities)
 - d. Impliciet idee is deels gebaseerd op expliciete kennis, maar ook voor een groot deel op ervaring
 - e. Momenteel een gebrek aan een taal om alle aspecten van deze processen goed te beschrijven (goed = alle belangrijke informatie bevatten en dus de impliciete kennis omzetten naar expliciete kennis)
 - f. Niet kunnen beschrijven betekent dat er weinig transparantie is, maar ook weinig homogeniteit in aangeboden dienst (+weinig geanalyseerd kan worden)
 - g. Hebben een taal gemaakt op basis van de geïdentificeerde noden van dit soort processen
 - h. Nu dus het resultaat evalueren (=taal, niet model!)
 - i. Project spoedgevallendienst AZMM in volgende fase
3. Beschrijven van de taal
 - a. Overlopen van het idee van de taal
 - i. Figuren imperatief vs declaratief
 - ii. Regels niet voldoende, beslissingen zijn centraal (=data)
Bv. wanneer bloedonderzoek?
 1. Preventief voor algemene gezondheidsstatus te bepalen
 2. Bij verdenking van ziekte
 - iii. Resources: beschikbaar personeel en materiaal
 - b. Overlopen van de soorten regels
 - i. 4 grote template groepen
 - ii. Voorbeelden van regels geven
4. Schetsen van het domein van de case
 - a. Dienst orthopedie: enkel diagnose en behandeling van patiënten met (vermoedelijke) botbreuken in vinger, hand, pols, onderarm, bovenarm, schouder en sleutelbeen
 - b. Oorspronkelijk volledig gebaseerd op online beschikbare info
5. Introduceren van het DeciClare tekstueel model van case
 - a. Beschrijving laten lezen
 - b. Model elementen overlopen
 - c. Betekenis van complexere regels uitleggen

B) Evaluatie Model

1. Semantische kwaliteit
 - a. Is het model correct?
 - b. Is het model volledig?
 - i. Amputatie?
 - ii. Medicatie?
 - iii. Andere tests (bv. voelen, mobiliteit testen...)?
 - iv. Andere behandelingen?
 - v. Andere beslissingen?
 - c. Zijn de voorbeelden conform met het model?
 - i. Ja
 - ii. Nee:

- Consult na X-ray
 - Rest na surgery
 - Cast na surgery
2. Pragmatische kwaliteit
 - a. Is het model nuttig ter ondersteuning van uw werkzaamheden?
 - b. Gebruikt u de informatie zoals in het model bij het nemen van beslissingen?
 - c. Hoe gebruikt u de beschreven kennis tijdens het uitvoeren van het proces? Is dit in termen van dit soort regels (impliciet of expliciet) of op een geheel andere wijze?
- C) Evaluatie Taal
1. Zou u iets kunnen bijleren van zulke modellen (bv. leggen van verbanden)?
 2. Denkt u dat de taal voldoende expressief is om andere soortgelijke processen te modelleren (bv. hele spoedgevallendienst)?
 3. Zijn er regels die u gebruikt of kennis van hebt in deze of andere gezondheidszorg processen, die u denkt dat niet met onze modelleertaal voor te stellen zijn?
 4. Vindt u het modelleren van dit soort processen met een dergelijke taal nuttig? Denkt u dat een meerwaarde kan zijn voor dokters of de gezondheidszorg in het algemeen?
- D) Einde

Appendix D: Final textual DeciClare model of arm fracture case

The formal model is available at:

github.com/stevmert/DeciClareBPMS/blob/master/engine/models/ArmFractureCase_DeciClareModel.txt

Defined Elements

- Available resources (13):
[nurse(OR nurse), doctor(anesthesiologist/orthopedic surgeon/emergency doctor), receptionist, surgical assistant, reception desk, examination room, X-ray room, CT room, operating room, patient room, patient bed, recovery room, recovery bed]
- Available activities (25):
[Register patient, Take medical history of patient, Clinical examination of patient, Doctor consultation, Take X-ray, Take CT scan, Prescribe NSAID painkillers, Prescribe SAID painkillers, Prescribe anticoagulants, Prescribe stomach protecting drug, Prescribe rest (=no treatment), Apply ice, Apply cast, Remove cast, Apply splint, Apply sling, Apply fixation, Remove fixation, Apply bandage, Apply figure of eight bandage, Perform surgery, Apply intra-compartmental pressure monitor, Let patient rest, Stay in patient room, Unregister patient]
- Available data records (2):
[Patient File, Real-time observations]
- Available data attributes (21):
[name=String, status=String, Patient has been to this hospital before?=(True/False), Is life-threatening emergency?=(True/False), Is conscious?=(True/False), Were there complications during surgery?=(True/False), Displaced fracture?=(True/False), Unstable fracture?=(True/False), Open fracture?=(True/False), More than 14yrs old?=(True/False), Bedridden patient?=(True/False), Operating time available?=(True/False), In pain?=(True/False), Is swollen?=(True/False), Numbness?=(True/False), Pin-and-needles?=(True/False), Has compartment syndrome?=(True/False), Has acute compartment syndrome?=(True/False), Needs surgery?=(True/False), Confirmed Fracture?=(Wrist/Forearm/Shoulder/Finger/Upper arm/Hand/Collarbone), Suspected fracture?=(Wrist/Forearm/Shoulder/Finger/Upper arm/Hand/Collarbone)]

Constraints

Data Constraints

- If[['Patient has been to this hospital before? = false']] then Register patient has to insert 'Patient File'
- If[['Patient has been to this hospital before? = false']] then Register patient has to insert 'Patient File->name'
- If[['Patient has been to this hospital before? = false']] then Register patient has to insert 'Patient File->status'
- If[['Patient has been to this hospital before? = true']] then Register patient has to read 'Patient File'
- If[['Patient has been to this hospital before? = false']] then Register patient has to read 'Patient File->name'
- If[['Patient has been to this hospital before? = true']] then Register patient has to update 'Patient File->status'
- Unregister patient has to update 'Patient File->status'

Resource Constraints

- there is at least 1 receptionist available
- there are at most 2 receptionists available
- receptionist can be used by up to 1 instance(s) simultaneously
- OPTIONAL there are at least 4 nurses available
- there are at most 10 nurses available
- nurse(OR nurse) can be used by up to 1 instance(s) simultaneously
- OPTIONAL there is at least 1 OR nurse available

- doctor(anesthesiologist/orthopedic surgeon/emergency doctor) can be used by up to 1 instance(s) simultaneously
- OPTIONAL there is at least 1 emergency doctor available
- there are at most 5 emergency doctors available
- OPTIONAL there is at least 1 anesthesiologist available
- there are at most 2 anesthesiologists available
- OPTIONAL there is at least 1 orthopedic surgeon available
- there are at most 3 orthopedic surgeons available
- OPTIONAL there is at least 1 surgical assistant available
- there are at most 4 surgical assistants available
- surgical assistant can be used by up to 1 instance(s) simultaneously
- there is at least 1 reception desk available
- there are at most 2 reception desks available
- reception desk can be used by up to 1 instance(s) simultaneously
- OPTIONAL there are at least 3 examination rooms available
- there are at most 15 examination rooms available
- examination room can be used by up to 1 instance(s) simultaneously
- there is at least 1 operating room available
- there are at most 3 operating rooms available
- operating room can be used by up to 1 instance(s) simultaneously
- OPTIONAL there is at least 1 X-ray room available
- there are at most 2 X-ray rooms available
- X-ray room can be used by up to 1 instance(s) simultaneously
- there is at most 1 CT room available
- CT room can be used by up to 1 instance(s) simultaneously
- OPTIONAL there are at least 2 recovery beds available
- there are at most 10 recovery beds available
- recovery bed can be used by up to 1 instance(s) simultaneously
- there is at least 1 recovery room available
- there is at most 1 recovery room available
- recovery room can be used by up to 10 instance(s) simultaneously
- OPTIONAL there are at least 5 patient beds available
- there are at most 100 patient beds available
- patient bed can be used by up to 1 instance(s) simultaneously
- there are at most 50 patient rooms available
- patient room can be used by up to 2 instance(s) simultaneously
- a receptionist has authorization over Register patient
- Register patient uses at least 1 receptionist
- Register patient uses at most 1 receptionist
- Register patient uses at least 1 reception desk
- Register patient uses at most 1 reception desk
- a receptionist has authorization over Unregister patient
- Unregister patient uses at least 1 receptionist
- Unregister patient uses at most 1 receptionist
- Unregister patient uses at least 1 reception desk
- Unregister patient uses at most 1 reception desk
- a doctor or nurse have authorization over Take medical history of patient
- Take medical history of patient uses at least 1 doctor or nurse

- a doctor has authorization over Clinical examination of patient
- If[['Is life-threatening emergency?' = false']] then Clinical examination of patient uses at least 1 examination room
- the decision of 'If[['Is life-threatening emergency?' = false']] then Clinical examination of patient uses at least 1 examination room' has to be taken by a doctor
- Clinical examination of patient uses at least 1 doctor
- a doctor has authorization over Doctor consultation
- Doctor consultation uses at least 1 examination room
- Doctor consultation uses at least 1 doctor
- a doctor has authorization over Take X-ray
- Take X-ray uses at least 1 X-ray room
- Take X-ray uses at least 1 doctor or nurse
- OPTIONAL Take X-ray uses at least 1 nurse
- a doctor has authorization over Take CT scan
- Take CT scan uses at least 1 CT room
- Take CT scan uses at most 1 CT room
- Take CT scan uses at least 1 doctor or nurse
- OPTIONAL Take CT scan uses at least 1 nurse
- a doctor has authorization over Apply cast
- Apply cast uses at least 1 examination room
- Apply cast uses at most 1 examination room
- Apply cast uses at least 1 doctor or nurse
- OPTIONAL Apply cast uses at least 1 nurse
- a doctor has authorization over Remove cast
- Remove cast uses at least 1 examination room
- Remove cast uses at most 1 examination room
- Remove cast uses at least 1 doctor or nurse
- OPTIONAL Remove cast uses at least 1 nurse
- a doctor has authorization over Apply fixation
- Apply fixation uses at least 1 operating room
- Apply fixation uses at most 1 operating room
- Apply fixation uses at least 1 orthopedic surgeon
- Apply fixation uses at least 1 scrub nurse or surgical assistant
- Apply fixation uses at least 1 anesthesiologist
- Apply fixation uses at least 1 OR nurse
- OPTIONAL Apply fixation uses at least 2 OR nurse
- a doctor has authorization over Remove fixation
- Remove fixation uses at least 1 operating room
- Remove fixation uses at most 1 operating room
- Remove fixation uses at least 1 orthopedic surgeon
- Remove fixation uses at least 1 scrub nurse or surgical assistant
- Remove fixation uses at least 1 anesthesiologist
- Remove fixation uses at least 1 OR nurse
- OPTIONAL Remove fixation uses at least 2 OR nurse
- a doctor has authorization over Apply figure of eight bandage
- Apply figure of eight bandage uses at least 1 examination room
- Apply figure of eight bandage uses at most 1 examination room
- Apply figure of eight bandage uses at least 1 doctor or nurse
- OPTIONAL Apply figure of eight bandage uses at least 1 nurse

- a doctor or nurse have authorization over Apply bandage
- Apply bandage uses at least 1 doctor or nurse
- OPTIONAL Apply bandage uses at least 1 nurse
- a doctor or nurse have authorization over Apply splint
- Apply splint uses at least 1 doctor or nurse
- OPTIONAL Apply splint uses at least 1 nurse
- a doctor or nurse have authorization over Apply sling
- Apply sling uses at least 1 doctor or nurse
- OPTIONAL Apply sling uses at least 1 nurse
- a doctor or nurse have authorization over Apply ice
- a doctor has authorization over Perform surgery
- Perform surgery uses at least 1 operating room
- Perform surgery uses at most 1 operating room
- Perform surgery uses at least 1 orthopedic surgeon
- Perform surgery uses at least 1 scrub nurse or surgical assistant
- Perform surgery uses at least 1 anesthesiologist
- Perform surgery uses at least 1 OR nurse
- OPTIONAL Perform surgery uses at least 2 OR nurse
- a doctor has authorization over Apply intra-compartmental pressure monitor
- Apply intra-compartmental pressure monitor uses at least 1 doctor or nurse
- a doctor has authorization over Let patient rest
- Let patient rest uses at least 1 recovery bed
- Let patient rest uses at most 1 recovery bed
- Let patient rest uses at least 1 recovery room
- Let patient rest uses at most 1 recovery room
- Stay in patient room uses at least 1 patient bed
- Stay in patient room uses at most 1 patient bed
- Stay in patient room uses at least 1 patient room
- Stay in patient room uses at most 1 patient room
- a doctor has authorization over Prescribe NSAID painkillers
- a doctor has authorization over Prescribe SAID painkillers
- a doctor has authorization over Prescribe anticoagulants
- a doctor has authorization over Prescribe stomach protecting drug
- a doctor has authorization over Prescribe rest (=no treatment)
- the following resources cannot be equal: orthopedic surgeon in Apply fixation or Perform surgery or Remove fixation, anesthesiologist in Apply fixation or Perform surgery or Remove fixation

Functional/Control-flow Constraints

- If[['Is life-threatening emergency? = false']] then (Register patient has to be performed at least 1 time) has to precede (Take medical history of patient has to be performed at least 1 time)
- If[['Is life-threatening emergency? = false']] then (Register patient has to be performed at least 1 time) has to precede (Clinical examination of patient has to be performed at least 1 time)
- If[['Is life-threatening emergency? = false']] then (Register patient has to be performed at least 1 time) has to precede (Doctor consultation has to be performed at least 1 time)
- If[['Is life-threatening emergency? = false']] then (Register patient has to be performed at least 1 time) has to precede (Take X-ray has to be performed at least 1 time)
- If[['Is life-threatening emergency? = false']] then (Register patient has to be performed at least 1 time) has to precede (Take CT scan has to be performed at least 1 time)

- If[['Is life-threatening emergency? = false']] then (Register patient has to be performed at least 1 time) has to precede (Prescribe NSAID painkillers has to be performed at least 1 time)
- If[['Is life-threatening emergency? = false']] then (Register patient has to be performed at least 1 time) has to precede (Prescribe SAID painkillers has to be performed at least 1 time)
- If[['Is life-threatening emergency? = false']] then (Register patient has to be performed at least 1 time) has to precede (Prescribe anticoagulants has to be performed at least 1 time)
- If[['Is life-threatening emergency? = false']] then (Register patient has to be performed at least 1 time) has to precede (Prescribe stomach protecting drug has to be performed at least 1 time)
- If[['Is life-threatening emergency? = false']] then (Register patient has to be performed at least 1 time) has to precede (Prescribe rest (=no treatment) has to be performed at least 1 time)
- If[['Is life-threatening emergency? = false']] then (Register patient has to be performed at least 1 time) has to precede (Apply ice has to be performed at least 1 time)
- If[['Is life-threatening emergency? = false']] then (Register patient has to be performed at least 1 time) has to precede (Apply cast has to be performed at least 1 time)
- If[['Is life-threatening emergency? = false']] then (Register patient has to be performed at least 1 time) has to precede (Remove cast has to be performed at least 1 time)
- If[['Is life-threatening emergency? = false']] then (Register patient has to be performed at least 1 time) has to precede (Apply splint has to be performed at least 1 time)
- If[['Is life-threatening emergency? = false']] then (Register patient has to be performed at least 1 time) has to precede (Apply sling has to be performed at least 1 time)
- If[['Is life-threatening emergency? = false']] then (Register patient has to be performed at least 1 time) has to precede (Apply fixation has to be performed at least 1 time)
- If[['Is life-threatening emergency? = false']] then (Register patient has to be performed at least 1 time) has to precede (Remove fixation has to be performed at least 1 time)
- If[['Is life-threatening emergency? = false']] then (Register patient has to be performed at least 1 time) has to precede (Apply bandage has to be performed at least 1 time)
- If[['Is life-threatening emergency? = false']] then (Register patient has to be performed at least 1 time) has to precede (Apply figure of eight bandage has to be performed at least 1 time)
- If[['Is life-threatening emergency? = false']] then (Register patient has to be performed at least 1 time) has to precede (Perform surgery has to be performed at least 1 time)
- If[['Is life-threatening emergency? = false']] then (Register patient has to be performed at least 1 time) has to precede (Apply intra-compartmental pressure monitor has to be performed at least 1 time)
- If[['Is life-threatening emergency? = false']] then (Register patient has to be performed at least 1 time) has to precede (Let patient rest has to be performed at least 1 time)
- If[['Is life-threatening emergency? = false']] then (Register patient has to be performed at least 1 time) has to precede (Stay in patient room has to be performed at least 1 time)
- If[['Is life-threatening emergency? = false']] then (Register patient has to be performed at least 1 time) has to precede (Unregister patient has to be performed at least 1 time)
- (Register patient has to be performed at least 1 time) has to be followed by (Unregister patient has to be performed at least 1 time)
- (Take medical history of patient has to be performed at least 1 time) has to be followed by (Unregister patient has to be performed at least 1 time)
- (Clinical examination of patient has to be performed at least 1 time) has to be followed by (Unregister patient has to be performed at least 1 time)
- (Doctor consultation has to be performed at least 1 time) has to be followed by (Unregister patient has to be performed at least 1 time)
- (Take X-ray has to be performed at least 1 time) has to be followed by (Unregister patient has to be performed at least 1 time)
- (Take CT scan has to be performed at least 1 time) has to be followed by (Unregister patient has to be performed at least 1 time)

- (Prescribe NSAID painkillers has to be performed at least 1 time) has to be followed by (Unregister patient has to be performed at least 1 time)
- (Prescribe SAID painkillers has to be performed at least 1 time) has to be followed by (Unregister patient has to be performed at least 1 time)
- (Prescribe anticoagulants has to be performed at least 1 time) has to be followed by (Unregister patient has to be performed at least 1 time)
- (Prescribe stomach protecting drug has to be performed at least 1 time) has to be followed by (Unregister patient has to be performed at least 1 time)
- (Prescribe rest (=no treatment) has to be performed at least 1 time) has to be followed by (Unregister patient has to be performed at least 1 time)
- (Apply ice has to be performed at least 1 time) has to be followed by (Unregister patient has to be performed at least 1 time)
- (Apply cast has to be performed at least 1 time) has to be followed by (Unregister patient has to be performed at least 1 time)
- (Remove cast has to be performed at least 1 time) has to be followed by (Unregister patient has to be performed at least 1 time)
- (Apply splint has to be performed at least 1 time) has to be followed by (Unregister patient has to be performed at least 1 time)
- (Apply sling has to be performed at least 1 time) has to be followed by (Unregister patient has to be performed at least 1 time)
- (Apply fixation has to be performed at least 1 time) has to be followed by (Unregister patient has to be performed at least 1 time)
- (Remove fixation has to be performed at least 1 time) has to be followed by (Unregister patient has to be performed at least 1 time)
- (Apply bandage has to be performed at least 1 time) has to be followed by (Unregister patient has to be performed at least 1 time)
- (Apply figure of eight bandage has to be performed at least 1 time) has to be followed by (Unregister patient has to be performed at least 1 time)
- (Perform surgery has to be performed at least 1 time) has to be followed by (Unregister patient has to be performed at least 1 time)
- (Apply intra-compartmental pressure monitor has to be performed at least 1 time) has to be followed by (Unregister patient has to be performed at least 1 time)
- (Let patient rest has to be performed at least 1 time) has to be followed by (Unregister patient has to be performed at least 1 time)
- (Stay in patient room has to be performed at least 1 time) has to be followed by (Unregister patient has to be performed at least 1 time)
- Register patient has to be performed at least 1 time
- Register patient can be performed at most 1 time
- Unregister patient has to be performed at least 1 time
- Unregister patient can be performed at most 1 time
- (Clinical examination of patient has to be performed at least 1 time) has to precede (Stay in patient room has to be performed at least 1 time)
- If[['Is conscious? = true']] then Take medical history of patient has to be performed at least 1 time
- the decision of 'If[['Is conscious? = true']] then Take medical history of patient has to be performed at least 1 time' has to be taken by a doctor or nurse
- Clinical examination of patient has to be performed at least 1 time
- If[['Is conscious? = true']] then (Take medical history of patient has to be performed at least 1 time) has to precede (Clinical examination of patient has to be performed at least 1 time)

- the decision of 'If[['Is conscious? = true']] then (Take medical history of patient has to be performed at least 1 time) has to precede (Clinical examination of patient has to be performed at least 1 time)' has to be taken by a doctor or nurse
- If[['Is life-threatening emergency? = false']] then Take X-ray has to be performed at least 1 time
- the decision of 'If[['Is life-threatening emergency? = false']] then Take X-ray has to be performed at least 1 time' has to be taken by a doctor
- If[['Suspected fracture? = Shoulder'] or ['Confirmed Fracture? = Shoulder' and 'Needs surgery? = false']] then Take CT scan has to be performed at least 1 time
- the decision of 'If[['Suspected fracture? = Shoulder'] or ['Confirmed Fracture? = Shoulder' and 'Needs surgery? = false']] then Take CT scan has to be performed at least 1 time' has to be taken by a doctor
- Take CT scan is only available [Every Monday, between 8am and 6pm]/[Every Tuesday, between 8am and 6pm]/[Every Wednesday, between 8am and 5pm]/[Every Thursday, between 8am and 6pm]/[Every Friday, between 8am and 6pm]
- If[['In pain? = true' and 'Is swollen? = true' and 'Pin-and-needles? = true' and 'Suspected fracture? = Hand'] or ['In pain? = true' and 'Suspected fracture? = Forearm' and 'Is swollen? = true' and 'Pin-and-needles? = true'] or ['In pain? = true' and 'Is swollen? = true' and 'Pin-and-needles? = true' and 'Suspected fracture? = Upper arm']] then Apply intra-compartmental pressure monitor has to be performed at least 1 time within at most 30 minutes
- the decision of 'If[['In pain? = true' and 'Is swollen? = true' and 'Pin-and-needles? = true' and 'Suspected fracture? = Hand'] or ['In pain? = true' and 'Suspected fracture? = Forearm' and 'Is swollen? = true' and 'Pin-and-needles? = true'] or ['In pain? = true' and 'Is swollen? = true' and 'Pin-and-needles? = true' and 'Suspected fracture? = Upper arm']] then Apply intra-compartmental pressure monitor has to be performed at least 1 time within at most 30 minutes' has to be taken by a doctor
- (Clinical examination of patient has to be performed at least 1 time) has to precede (Doctor consultation has to be performed at least 1 time)
- (Clinical examination of patient has to be performed at least 1 time) has to precede (Take X-ray has to be performed at least 1 time)
- (Clinical examination of patient has to be performed at least 1 time) has to precede (Take CT scan has to be performed at least 1 time)
- (Clinical examination of patient has to be performed at least 1 time) has to precede (Apply intra-compartmental pressure monitor has to be performed at least 1 time)
- (Take CT scan has to be performed at least 1 time) or (Take X-ray has to be performed at least 1 time) have to be followed by (Doctor consultation has to be performed at least 1 time)
- at least 1 out of 11 of the following activities have to be performed: Prescribe SAID painkillers or Prescribe rest (=no treatment) or Apply sling or Apply figure of eight bandage or Apply fixation or Apply ice or Prescribe NSAID painkillers or Perform surgery or Apply splint or Apply cast or Apply bandage
- (Clinical examination of patient has to be performed at least 1 time) has to precede (Prescribe NSAID painkillers has to be performed at least 1 time)
- (Clinical examination of patient has to be performed at least 1 time) has to precede (Prescribe SAID painkillers has to be performed at least 1 time)
- (Clinical examination of patient has to be performed at least 1 time) has to precede (Prescribe anticoagulants has to be performed at least 1 time)
- (Clinical examination of patient has to be performed at least 1 time) has to precede (Prescribe stomach protecting drug has to be performed at least 1 time)
- (Clinical examination of patient has to be performed at least 1 time) has to precede (Prescribe rest (=no treatment) has to be performed at least 1 time)
- (Clinical examination of patient has to be performed at least 1 time) has to precede (Apply cast has to be performed at least 1 time)
- (Clinical examination of patient has to be performed at least 1 time) has to precede (Apply fixation has to be performed at least 1 time)

- (Clinical examination of patient has to be performed at least 1 time) has to precede (Apply figure of eight bandage has to be performed at least 1 time)
- (Clinical examination of patient has to be performed at least 1 time) has to precede (Perform surgery has to be performed at least 1 time)
- OPTIONAL If[['Is swollen? = true'] or ['Has compartment syndrome? = true']] then Apply ice has to be performed at least 1 time unless If[['Is swollen? = false' and 'Has compartment syndrome? = false']]
- the decision of 'OPTIONAL If[['Is swollen? = true'] or ['Has compartment syndrome? = true']] then Apply ice has to be performed at least 1 time unless If[['Is swollen? = false' and 'Has compartment syndrome? = false']] has to be taken by a doctor
- OPTIONAL before (Prescribe anticoagulants has to be performed at least 1 time) can be executed after the execution of (Prescribe NSAID painkillers has to be performed at least 1 time) at least 12 hours have to have elapsed
- OPTIONAL before (Prescribe NSAID painkillers has to be performed at least 1 time) can be executed after the execution of (Prescribe anticoagulants has to be performed at least 1 time) at least 12 hours have to have elapsed
- OPTIONAL (Prescribe anticoagulants has to be performed at least 1 time) has to be directly followed by (Prescribe stomach protecting drug has to be performed at least 1 time)
- OPTIONAL If[['In pain? = true']] then Prescribe SAID painkillers or Prescribe NSAID painkillers have to be performed at least a combined 1 time
- the decision of 'OPTIONAL If[['In pain? = true']] then Prescribe SAID painkillers or Prescribe NSAID painkillers have to be performed at least a combined 1 time' has to be taken by a doctor
- (Apply cast has to be performed at least 1 time) has to be followed by (Remove cast has to be performed at least 1 time)
- (Apply cast has to be performed at least 1 time) has to precede (Remove cast has to be performed at least 1 time)
- (Apply fixation has to be performed at least 1 time) has to be followed by (Remove fixation has to be performed at least 1 time)
- (Apply fixation has to be performed at least 1 time) has to precede (Remove fixation has to be performed at least 1 time)
- If[['Has compartment syndrome? = true']] then (Apply cast has to be performed at least 1 time) and (Remove cast can be performed at most 0 times) have to be followed by (Remove cast has to be performed at least 1 time) within at most 15 minutes
- the decision of 'If[['Has compartment syndrome? = true']] then (Apply cast has to be performed at least 1 time) and (Remove cast can be performed at most 0 times) have to be followed by (Remove cast has to be performed at least 1 time) within at most 15 minutes' has to be taken by a doctor
- If[['Has compartment syndrome? = true']] then (Apply fixation has to be performed at least 1 time) and (Remove fixation can be performed at most 0 times) have to be followed by (Remove fixation has to be performed at least 1 time) within at most 15 minutes
- the decision of 'If[['Has compartment syndrome? = true']] then (Apply fixation has to be performed at least 1 time) and (Remove fixation can be performed at most 0 times) have to be followed by (Remove fixation has to be performed at least 1 time) within at most 15 minutes' has to be taken by a doctor
- If[['Displaced fracture? = true' and 'Operating time available? = false' and 'Confirmed Fracture? = Wrist'] or ['Displaced fracture? = true' and 'Operating time available? = false' and 'Confirmed Fracture? = Forearm']] then ((Apply cast has to be performed at least 1 time) or (Apply splint has to be performed at least 1 time)) and (Stay in patient room has to be performed at least 1 time) have to precede (Perform surgery has to be performed at least 1 time) unless If[['Operating time available? = true']]
- the decision of 'If[['Displaced fracture? = true' and 'Operating time available? = false' and 'Confirmed Fracture? = Wrist'] or ['Displaced fracture? = true' and 'Operating time available? = false' and 'Confirmed Fracture? = Forearm']] then ((Apply cast has to be performed at least 1 time) or (Apply splint has to be performed at least 1 time))

- 1 time)) and (Stay in patient room has to be performed at least 1 time) have to precede (Perform surgery has to be performed at least 1 time) unless If[['Operating time available? = true']] has to be taken by a doctor
- If[['Confirmed Fracture? = Shoulder'] or ['Confirmed Fracture? = Upper arm'] or ['Confirmed Fracture? = Forearm'] or ['Confirmed Fracture? = Wrist'] or ['Confirmed Fracture? = Finger'] or ['Confirmed Fracture? = Hand']] then Apply sling has to be performed at least 1 time
 - the decision of 'If[['Confirmed Fracture? = Shoulder'] or ['Confirmed Fracture? = Upper arm'] or ['Confirmed Fracture? = Forearm'] or ['Confirmed Fracture? = Wrist'] or ['Confirmed Fracture? = Finger'] or ['Confirmed Fracture? = Hand']] then Apply sling has to be performed at least 1 time' has to be taken by a doctor or nurse
 - If[['Open fracture? = false' and 'Confirmed Fracture? = Forearm' and 'Displaced fracture? = false'] or ['Confirmed Fracture? = Shoulder' and 'Open fracture? = false' and 'Displaced fracture? = false'] or ['Open fracture? = false' and 'More than 14yrs old? = true' and 'Displaced fracture? = false' and 'Confirmed Fracture? = Wrist']] then Apply cast or Apply splint have to be performed at least a combined 1 time
 - the decision of 'If[['Open fracture? = false' and 'Confirmed Fracture? = Forearm' and 'Displaced fracture? = false'] or ['Confirmed Fracture? = Shoulder' and 'Open fracture? = false' and 'Displaced fracture? = false'] or ['Open fracture? = false' and 'More than 14yrs old? = true' and 'Displaced fracture? = false' and 'Confirmed Fracture? = Wrist']] then Apply cast or Apply splint have to be performed at least a combined 1 time' has to be taken by a doctor
 - If[['Confirmed Fracture? = Finger' and 'Unstable fracture? = false' and 'Open fracture? = false' and 'Displaced fracture? = false']] then Apply splint and Apply bandage have to be performed at least a combined 1 time
 - the decision of 'If[['Confirmed Fracture? = Finger' and 'Unstable fracture? = false' and 'Open fracture? = false' and 'Displaced fracture? = false']] then Apply splint and Apply bandage have to be performed at least a combined 1 time' has to be taken by a doctor
 - If[['Confirmed Fracture? = Finger' and 'Open fracture? = true'] or ['Confirmed Fracture? = Finger' and 'Displaced fracture? = true'] or ['Confirmed Fracture? = Finger' and 'Unstable fracture? = true']] then Apply fixation has to be performed at least 1 time
 - the decision of 'If[['Confirmed Fracture? = Finger' and 'Open fracture? = true'] or ['Confirmed Fracture? = Finger' and 'Displaced fracture? = true'] or ['Confirmed Fracture? = Finger' and 'Unstable fracture? = true']] then Apply fixation has to be performed at least 1 time' has to be taken by a doctor
 - If[['Unstable fracture? = false' and 'Open fracture? = false' and 'Displaced fracture? = false' and 'Confirmed Fracture? = Hand']] then Apply fixation or Apply splint or Apply cast have to be performed at least a combined 1 time
 - the decision of 'If[['Unstable fracture? = false' and 'Open fracture? = false' and 'Displaced fracture? = false' and 'Confirmed Fracture? = Hand']] then Apply fixation or Apply splint or Apply cast have to be performed at least a combined 1 time' has to be taken by a doctor
 - If[['Unstable fracture? = false' and 'Confirmed Fracture? = Upper arm' and 'Open fracture? = false' and 'Displaced fracture? = false']] then Apply sling or Apply cast or Apply splint have to be performed at least a combined 1 time
 - the decision of 'If[['Unstable fracture? = false' and 'Confirmed Fracture? = Upper arm' and 'Open fracture? = false' and 'Displaced fracture? = false']] then Apply sling or Apply cast or Apply splint have to be performed at least a combined 1 time' has to be taken by a doctor
 - If[['Unstable fracture? = false' and 'Open fracture? = false' and 'Displaced fracture? = false' and 'Confirmed Fracture? = Collarbone']] then Apply figure of eight bandage or Apply splint have to be performed at least a combined 1 time
 - the decision of 'If[['Unstable fracture? = false' and 'Open fracture? = false' and 'Displaced fracture? = false' and 'Confirmed Fracture? = Collarbone']] then Apply figure of eight bandage or Apply splint have to be performed at least a combined 1 time' has to be taken by a doctor
 - OPTIONAL If[['Confirmed Fracture? = Collarbone']] then Apply sling has to be performed at least 1 time
 - the decision of 'OPTIONAL If[['Confirmed Fracture? = Collarbone']] then Apply sling has to be performed at least 1 time' has to be taken by a doctor or nurse

- If[['Is life-threatening emergency? = false' and 'Suspected fracture? = Shoulder']] then Take X-ray or Take CT scan have to be performed at least a combined 1 time
- the decision of 'If[['Is life-threatening emergency? = false' and 'Suspected fracture? = Shoulder']] then Take X-ray or Take CT scan have to be performed at least a combined 1 time' has to be taken by a doctor
- If[['Confirmed Fracture? = Shoulder']] then Take X-ray has to be performed at least 1 time after at least 1 day
- the decision of 'If[['Confirmed Fracture? = Shoulder']] then Take X-ray has to be performed at least 1 time after at least 1 day' has to be taken by a doctor
- OPTIONAL If[['Confirmed Fracture? = Shoulder' and 'Open fracture? = false' and 'Displaced fracture? = false']] then Perform surgery can be performed at most 0 times
- the decision of 'OPTIONAL If[['Confirmed Fracture? = Shoulder' and 'Open fracture? = false' and 'Displaced fracture? = false']] then Perform surgery can be performed at most 0 times' has to be taken by a doctor
- If[['Were there complications during surgery? = false']] then (Perform surgery has to be performed at least 1 time) has to be followed by (Let patient rest has to be performed at least 1 time) within at most 5 minutes
- the decision of 'If[['Were there complications during surgery? = false']] then (Perform surgery has to be performed at least 1 time) has to be followed by (Let patient rest has to be performed at least 1 time) within at most 5 minutes' has to be taken by a doctor
- If[['Displaced fracture? = true'] or ['Open fracture? = true']] then Perform surgery has to be performed at least 1 time
- the decision of 'If[['Displaced fracture? = true'] or ['Open fracture? = true']] then Perform surgery has to be performed at least 1 time' has to be taken by a doctor
- If[['Has acute compartment syndrome? = true']] then Perform surgery has to be performed at least 1 time within at most 1 hour
- the decision of 'If[['Has acute compartment syndrome? = true']] then Perform surgery has to be performed at least 1 time within at most 1 hour' has to be taken by a doctor
- If[['Unstable fracture? = true']] then Perform surgery or (Take X-ray and Doctor consultation) have to be performed at least a combined 1 time
- the decision of 'If[['Unstable fracture? = true']] then Perform surgery or (Take X-ray and Doctor consultation) have to be performed at least a combined 1 time' has to be taken by a doctor
- (Perform surgery has to be performed at least 1 time) has to be followed by (Prescribe NSAID painkillers has to be performed at least 1 time) or ((Prescribe SAID painkillers has to be performed at least 1 time) and (Prescribe anticoagulants has to be performed at least 1 time)) within at most 4 hours
- If[['Bedridden patient? = true']] then (Perform surgery has to be performed at least 1 time) has to be followed by (Prescribe SAID painkillers has to be performed at least 1 time) and (Prescribe anticoagulants has to be performed at least 1 time) within at most 4 hours
- the decision of 'If[['Bedridden patient? = true']] then (Perform surgery has to be performed at least 1 time) has to be followed by (Prescribe SAID painkillers has to be performed at least 1 time) and (Prescribe anticoagulants has to be performed at least 1 time) within at most 4 hours' has to be taken by a doctor
- (Apply cast has to be performed at least 1 time) has to be followed by (Perform surgery has to be performed at least 1 time) or (Take X-ray has to be performed at least 1 time)
- If[['Open fracture? = true' and 'Confirmed Fracture? = Forearm'] or ['Open fracture? = true' and 'Confirmed Fracture? = Wrist'] or ['Displaced fracture? = true' and 'Confirmed Fracture? = Wrist'] or ['Displaced fracture? = true' and 'Confirmed Fracture? = Forearm']] then (Perform surgery has to be performed at least 1 time) has to be followed by (Apply cast has to be performed at least 1 time) or (Apply splint has to be performed at least 1 time)
- the decision of 'If[['Open fracture? = true' and 'Confirmed Fracture? = Forearm'] or ['Open fracture? = true' and 'Confirmed Fracture? = Wrist'] or ['Displaced fracture? = true' and 'Confirmed Fracture? = Wrist'] or ['Displaced fracture? = true' and 'Confirmed Fracture? = Forearm']] then (Perform surgery has to be performed at least 1 time) has to be followed by (Apply cast has to be performed at least 1 time) or (Apply splint has to be performed at least 1 time)' has to be taken by a doctor

- If[['Is life-threatening emergency? = false' and 'Suspected fracture? = Hand'] or ['Suspected fracture? = Finger' and 'Is life-threatening emergency? = false'] or ['Is life-threatening emergency? = false' and 'Suspected fracture? = Upper arm'] or ['Is life-threatening emergency? = false' and 'Suspected fracture? = Collarbone'] or ['Is life-threatening emergency? = false' and 'Suspected fracture? = Forearm'] or ['Is life-threatening emergency? = false' and 'Suspected fracture? = Wrist']] then (Take X-ray has to be performed at least 1 time) has to precede (Apply fixation has to be performed at least 1 time) or (Apply cast has to be performed at least 1 time) or (Apply figure of eight bandage has to be performed at least 1 time) or (Perform surgery has to be performed at least 1 time)
- the decision of 'If[['Is life-threatening emergency? = false' and 'Suspected fracture? = Hand'] or ['Suspected fracture? = Finger' and 'Is life-threatening emergency? = false'] or ['Is life-threatening emergency? = false' and 'Suspected fracture? = Upper arm'] or ['Is life-threatening emergency? = false' and 'Suspected fracture? = Collarbone'] or ['Is life-threatening emergency? = false' and 'Suspected fracture? = Forearm'] or ['Is life-threatening emergency? = false' and 'Suspected fracture? = Wrist']] then (Take X-ray has to be performed at least 1 time) has to precede (Apply fixation has to be performed at least 1 time) or (Apply cast has to be performed at least 1 time) or (Apply figure of eight bandage has to be performed at least 1 time) or (Perform surgery has to be performed at least 1 time)' has to be taken by a doctor
- OPTIONAL If[['In pain? = true']] then (Prescribe anticoagulants can be performed at most 0 times) and ((Doctor consultation has to be performed at least 1 time) or (Clinical examination of patient has to be performed at least 1 time) or (Perform surgery has to be performed at least 1 time)) have to be directly followed by (Prescribe NSAID painkillers has to be performed at least 1 time)
- the decision of 'OPTIONAL If[['In pain? = true']] then (Prescribe anticoagulants can be performed at most 0 times) and ((Doctor consultation has to be performed at least 1 time) or (Clinical examination of patient has to be performed at least 1 time) or (Perform surgery has to be performed at least 1 time)) have to be directly followed by (Prescribe NSAID painkillers has to be performed at least 1 time)' has to be taken by a doctor

Appendix E: The join step of DeciClareMiner (decision-independent)

Rules with a logical expression of size k are joined to create rules with a logical expression of size $k+1$ in the iteration steps of the algorithm. However, the elements of the logical expressions can be connected with either the inclusive disjunction or the logical conjunction operator. Therefore, the influence of joining these logical expressions using either operator on the number of violations and the support level has been investigated for each template to apply the leveraged property.

The relation constraint templates have two possible logical expressions that can be joined: joining the logical expressions on the condition sides of relation constraints and joining the logical expressions on the consequence sides. Table 25 and Table 26 present the lower and upper bound for the number of conforming ($\#C$) and violating ($\#V$) occurrences of the child rule relative to the respective values for the parent rules for each type of join operation. For example, consider rules $\text{Response}(\text{AtLeast}(A, 1), \text{AtLeast}(B, 1))$ and $\text{Response}(\text{AtLeast}(A, 1), \text{AtLeast}(C, 1))$ for an event log consisting of four times trace ‘AB’ and two times ‘AC’. The former rule has 4 confirming and 2 violating occurrences, while the latter has 2 confirming and 4 violating occurrences. Table 26 states that the rule resulting from the OR-join operation on the consequence side, $\text{Response}(\text{AtLeast}(A, 1), \text{AtLeast}(B \text{ or } C, 1))$, will have at least 4 ($=\max[4, 2]$) and at most 6 ($=\min[4+2, 6]$) confirming occurrences and at least 0 ($=\max[6-6, 0]$) and at most 2 ($=\min[4, 2]$) violating occurrences. Evaluating the rule results in 6 confirming occurrences and no violations.

A distinction is also made between the positive (i.e., ‘at least ...’ consequence) and negative (i.e., ‘at most 0’ consequence) versions of the relation templates, as this allows for more specific bounds.

	<i>‘At least 1’ consequence expression</i>		<i>‘At most 0’ consequence expression</i>	
	<i>OR-join</i>	<i>AND-join</i>	<i>OR-join</i>	<i>AND-join</i>
Max #C	sum($\#C$)	min($\#C$)	sum($\#C$)	min($\#C$)
Min #C	max($\#C$)	0	max($\#C$)	0
Max #V	sum($\#V$)	min($\#V$)	sum($\#V$)	min($\#V$)
Min #V	max($\#V$)	0	max($\#V$)	0

Table 25. The bounds for join operation of the condition side of the relation templates

	<i>‘At least 1’ consequence expression</i>		<i>‘At most 0’ consequence expression</i>	
	<i>OR-join</i>	<i>AND-join</i>	<i>OR-join</i>	<i>AND-join</i>
Max #C	min(sum($\#C$), #Occurrences)	min($\#C$)	min($\#C$)	min(sum($\#C$), #Occurrences)
Min #C	max($\#C$)	max(sum($\#C$) - #Occurrences, 0)	max(sum($\#C$) - #Occurrences, 0)	max($\#C$)
Max #V	min($\#V$)	min(sum($\#V$), #Occurrences)	min(sum($\#V$), #Occurrences)	min($\#V$)
Min #V	max(sum($\#V$) - #Occurrences, 0)	max($\#V$)	max($\#V$)	max(sum($\#V$) - #Occurrences, 0)

Table 26. The bounds for join operation of the consequence side of the relation templates

Based on Table 25 and Table 26, optimizations can be formulated for the join operations for each type of relation rule. At the condition side, it is clear that the OR-join will never result in a minimal rule as the number of violations of the child rule is at least equal to the highest number of violations of its parent rules. So only the AND-join operation will be used when joining condition sides. Additionally, these bounds make it possible to know beforehand if the resulting rule can have a support level higher than the threshold. The maximum support level of the child rule is the sum of the maximal number of conforming occurrences and the maximal number of violating occurrences. In this case, this is equal to the sum of the minimal number of conforming occurrences and the minimal number of violating occurrences of the parent rules. A similar conclusion can be drawn at the consequence side, however, the join operation to use now depends on whether or not the constraint in question is based on a positive or a negative relation template. Positive constraints will only use the OR-join, while negative constraints will only use the AND-join. The bounds for the consequence side do not allow for any conclusions to be drawn about the support level of the child rule.

Additionally, the iteration step will be executed twice for relation templates, because the join operations are defined for the condition and consequence side separately. The first time it will be executed with the input described in the initialization step. This will only grow the size of the condition side logical expression of the constraints. The resulting output is a set of all the minimal rules that occur at least once with a consequence side logical expression of length one. This output is then again used as input for the algorithm, but this time the consequence side logical expression is the one that is allowed to grow.

The proposed algorithm only supports two of the resource templates, namely the `AtLeastUsage1` and `AtMostUsage0` templates, as the others are difficult to mine from traditional event logs. For these two templates, the bounds for the number of confirming and violating occurrences of this join operation are the same as for the positive relation templates in Table 25.

Appendix F: The DeciClare templates supported by DeciClareEngine

The current implementation of the DeciClareEngine-tool supports the 18 most important constraint templates of DeciClare. Support for more templates will be added later. Table 27 presents the currently supported constraint templates, each of which is accompanied by an example that illustrates its meaning and parameters. The parameters I and B define the logical operators that can be used in the parameter expressions of the template because of branching. The letter used refers to the option of using either the inclusive disjunction (I) or both the inclusive disjunction and conjunction (B).

Although supported by the DeciClareEngine, Table 27 omits the timing parameters required for each of the templates, except for the untimed RespondedPresence, AtLeastLag and ActivityAuthorization templates, to avoid overloading the table. These are two parameters expressing the time interval in which the constraint applies. For example, `Response(AtLeast(e1, 1), AtLeast(e2, 1), 2, 7)` expresses that if `e1` is executed, then `e2` must be executed at least once after at least 2 time units and at most 7 time units after `e1` was executed. The lower- and upper-bound time parameters can be 0 and infinity, respectively, reducing them to the examples presented in Table 27.

Similarly, activation decisions are omitted in Table 27 but supported by DeciClareEngine. The deactivation decisions are not yet supported.

Class	Template	Example
<i>Existence</i>	AtLeast(I, n)	AtLeast(Apply bandage or Apply cast, 2): the sum of the occurrences of Apply bandage and the occurrences of Apply cast has to be at least 2
	AtMost(B, n)	AtMost(Examine patient and Take X-ray, 3): the number of co-occurrences of Examine patient and Take X-ray must be at most 3
	AtLeastChoice(B, n)	AtLeastChoice(Prescribe SAID painkillers or Prescribe NSAID painkillers, 1): at least one of the two activities has to be executed (independent of how many times)
	AtMostChoice(B, n)	AtMostChoice(Prescribe SAID painkillers or Prescribe anticoagulants, 1): at most one of the two activities can to be executed (independent of how many times)
	First(I)	First(Register patient, Examine patient): either Register patient or Examine patient has to be executed before any other event can be executed
	Last(I)	Last(Apply sling or Examine patient): either Apply sling or Examine patient has to be executed as the final event
<i>Relation</i>	RespondedPresence(B ₁ , B ₂)	RespondedPresence(AtLeast((Take X-ray and Apply sling) or Apply bandage, 1), AtLeast(Examine patient or Perform surgery, 1)): if Take X-ray and Apply sling or just Apply bandage are executed at least once, then Examine patient or Perform surgery must also be executed at least once somewhere in the instance (before, during or after)
	Response(B ₁ , B ₂)	Response(AtLeast((Take X-ray and Apply sling) or Apply bandage, 1), AtLeast(Examine patient or Perform surgery, 1)): if Take X-ray and Apply sling or just Apply bandage are executed at least once, then Examine patient or Perform surgery must also be executed at least once at some time after Take X-ray and Apply sling or after Apply bandage
	ChainResponse(B ₁ , B ₂)	ChainResponse(AtLeast((Take X-ray and Apply sling) or Apply bandage, 1), AtLeast(Examine patient or Perform surgery, 1)): if Take X-ray and Apply sling or just Apply bandage are/is executed at least once, then Examine patient or Perform surgery must also be executed at least once immediately after Take X-ray and Apply sling or after Apply bandage
	Precedence(B ₁ , B ₂)	Precedence(AtLeast((Take X-ray and Apply sling) or Apply bandage, 1), AtLeast(Examine patient or Perform surgery, 1)): Take X-ray and Apply sling or just Apply bandage can be executed only after Examine patient or Perform surgery has already been executed at least once
	ChainPrecedence(B ₁ , B ₂)	ChainPrecedence(AtLeast((Take X-ray and Apply sling) or Apply bandage, 1), AtLeast(Examine patient or Perform surgery, 1)): Take X-ray and Apply sling or just Apply bandage can be executed only immediately after Examine patient or Perform surgery has been executed at least once
	AtLeastLag(B ₁ , B ₂ , n)	AtLeastLag(AtLeast(Give painkiller, 1), AtLeast(Take anticoagulants, 1), 12): if Give painkiller is executed at least once, then at least 12 time units have to have passed before Take anticoagulants can be executed at least once
<i>Resource</i>	AtLeastAvailable(I, n)	AtLeastAvailable(nurse, 2): there should be at least 2 nurses available to the process (but they are allowed to be in use by the process).
	AtMostAvailable(I, n)	AtMostAvailable(nurse, 4): there should be no more than 4 nurses available to the process
	AtLeastUsage(act, I, n)	AtLeastUsage(Register patient, receptionist, 1): at least 1 receptionist is needed to execute Register patient
	AtMostUsage(act, I, n)	AtMostUsage(Register patient, receptionist, 2): no more than 2 receptionists should be used to execute Register patient
	(No)ActivityAuthorization(I, act)	ActivityAuthorization(doctor, Clinical exam): a doctor is allowed to execute a Clinical exam

Table 27. The constraint templates supported by DeciClareEngine