Eindhoven University of Technology

MASTER

A performance measurement framework for executable care pathways implemented in BPMS

Gita Gustaman, G.

Award date:
2014

Link to publication

Disclaimer
This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 17. Mar. 2025

Eindhoven University of Technology
Department of Industrial Engineering and Innovation Sciences
Department of Mathematics and Computer Science

A Performance Measurement Framework for Executable Care Pathways Implemented in BPMS

by
Goppy Gita Gustaman
Student ID: 0827042

In partial fulfilment of the requirements for the degree of
Master of Science in Business Information Systems

Eindhoven, October 2014

Supervisor 1: Prof. Dr. Ir. Uzay Kaymak (TU/e, IE&IS)
Supervisor 2: Dr. P.M.E. Van Gorp (TU/e, IE&IS)

Acknowledgement

This master thesis is the result of my graduation project fulfilled in order to obtain the degree of Master of Science in Business Information Systems. The research was carried out at the Information Systems group, Department of Industrial Engineering and Innovation Sciences, Eindhoven University of Technology.

I would like to thank to a few people who have contributed to this research. First of all, I would like to thank my first supervisor, Uzay Kaymak for his guidance and his brain teasing feedback to improve my research. I would also like to thank Pieter Van Gorp as my second supervisor for his valuable input. I also thank Hui Yan for our discussion about weaning process and the model. I thank Irene Vanderfeesten for her feedback about weaning process. I also thank Alexander Serebrenik for his input about domain specific language. Many thanks go to JBoss Community for your active forum; I can solve some technical difficulties.

Second, I would like to thank my friends. Special thank goes to Juby Joseph Ninan for our intensive discussion during the research. Many thanks go out to my fellow students: Kostas, Igor, Ekaterina, Yi, Pavel and Kritika for your supports and these wonderful two years.

Third, I would like to thank PT Petrokimia Gresik who supports me morally and financially. Last, but not least, I would like to express my gratitude to my family: my parents, my wife Tatya and my daughter Hasna for always being there for me.

Goppy Gita Gustaman
October 2014

Abstract

Care pathways are developed to standardize care processes for a well-defined group of patients during a well-defined period. It is implemented in various forms, such as paper based version and electronic version in the form of information systems. Several works propose the implementation of a care pathway as a workflow application executed in a business process management system (BPMS) due to its strength in work routing and the opportunity for process improvement.

Like any other business processes, continuous performance monitoring of care pathway is important to assess how far the organization's goals are achieved while promoting the patient safety principle and improving the quality of care. However, care pathway performance measurement activity has several practical issues, such as unavailability of performance measurement system, distribution of data in multiple locations and performance indicators that are expressed in non-executable natural language. Consequently, performance monitoring is difficult to achieve and time consuming.

Motivated by the promising implementation of care pathway in BPMS and the necessity to continuously monitor pathway implementation, this master project proposes a framework to measure the performance of care pathways that are implemented in BPMS. A pathway-specific, structured notation is introduced to formalize the performance indicators of care pathways. The notation is designed to be executable against BPMS's execution log that represents patient record in order to calculate indicator values. In this project, a BPMS is selected to implement the pathway, i.e. jBPM. The performance indicator is formalized on top of identified performance measures as the result of analysis of jBPM logs and a care pathway performance measurement guideline, i.e. the Leuven Clinical Pathway Compass. To prove the concept of indicator formalization, a performance measurement system is developed, thus automating the measurement. To assess the quality of the proposed approach, the formalization and the measurement system are evaluated using a healthcare case, i.e. weaning process. The evaluation concludes that the formalization can support performance monitoring of the case and is open for further extension. This master project demonstrates that continuous performance monitoring of care pathway is feasible using proposed framework.

Keywords: care pathway, performance measurement, performance indicator, domain specific language, jBPM

Contents

List of Figures

List of Tables

List of Scenarios

Chapter 1

Introduction

Healthcare sector is facing challenges to improve its quality while reducing the cost at the same time. This is due to the dynamicity in the patient situation such as aging population and chronic illness (Deloitte, 2014). Healthcare business is also pressured where the rise in medical expenses is the matter while maintaining its service quality. Moreover, there is a force from policymakers and third party such as insurance company to hospitals to deliver their care services in most effective and efficient way.

There are already several researches conducted to tackle effectiveness and efficiency issues where process orientation approach is at the utmost attention. Care pathway (CP) is an example of such approach applied to standardize care process (Vanhaecht, K.; De Witte, K.; Sermeus, W., 2007). The European Pathway Association defines a care pathway, also known as clinical pathway, as "*a complex intervention for the mutual decision making and organisation of care processes for a well-defined group of patients during a well-defined period* (European Pathway Association, 2014).

Care pathway implementation evolves from paper version in the mid-1980s, electronic version of paper based and linear sequential model in the early 1990s, and state-transition model in the late 1990s. Since 2000, there have been many researches on structural design of clinical pathway (Wakamiya & Yamauchi, 2009). At the same time, the evolution of information technology and science is also on the way. Several proposals were made to facilitate care pathway design and implementation using information technology and science concept. Ye et al proposed ontology-based approach to model care pathways workflows to facilitate computerized implementation of it (Ye, Jiang, Diao, Yang, & Du, 2009). Business process management (BPM) approach, which is a buzzword in recent years, was also adopted to design a care pathway as a process model. Du et al proposes a framework to implement CP based on workflow in Petri net (Du, G., Jiang, Z., & Diao, X., 2008). Vermeulen proposes a methodology to model a care pathway as a process model (Vermeulen, 2013). In the same year, Renswouw also proposes a methodology to convert paper-based care pathway into executable workflow process model (van Renswouw, 2013). Both Vermeulen and Renswouw use BPMN 2.0 as the process modeling language.

Continuous business performance monitoring is important to see how far a business achieving its goals and hence a correct and quick action can be taken by decision makers in case deviation occurs. In care pathway case, continuous performance monitoring and follow up are crucial to ensure quality of care while promoting patient safety. In addition, the hospitals can benefit from measurement result to see if they comply with law, standard, and regulation. Unfortunately,

despite many proposals on care pathway structural design, there is a lack of further researches on the performance measurement technique given specific structure of care pathway.

1.1 Problem Definition

The promising adoption of BPM in healthcare sector is the driver to implement care pathway as a process model and bring it into an executable system, as shown by several works (Du, G., Jiang, Z., & Diao, X., 2008) (Vermeulen, 2013) (van Renswouw, 2013). In BPM lifecycle, the next step of business process implementation is business process monitoring (Dumas, La Rosa, Mendling, & Reijers, 2013). The essence of monitoring is basically measuring performance which employs several (key) performance indicators (KPI). Those indicators can be derived from guidelines, either from generic business performance monitoring guidelines such as Balanced Scorecard or from domain specific guidelines, such as The Leuven Clinical Pathway Compass for care pathway.

The problem lies in bringing the guideline into the practice. The availability and distribution of data, availability of automatic measurement system and performance indicators which are expressed in natural language are sort of examples. Thus a bridge is required between care pathway performance measurement guideline and its implementation in a BPM system. This is important to make sure that the BPM system is able to facilitate performance measurement for care pathway.

In many organizations including healthcare sector, performance indicators are expressed in natural language which is unstructured. This has several drawbacks. Besides not machine executable, natural language can lead to ambiguity. Consequently, it is resource consuming to perform measurement and will result in high error rate. Moreover, performance indicators are object that are communicated across functions within organization. For healthcare environment, it is communicated for example among business owner, medical team and IT team as the medical data manager. The problem above leads to the necessity to develop an approach to formalize care pathway performance indicator that is easily understood by multiple functions. In addition, it is also necessary to develop a system to facilitate easy, automatic, and high quality measurement.

There is an increasing interest in open source software in general and BPM system(BPMS) in particular (Wohed, Russel, ter Hofstede, Andersson, & van der Aalst, 2008). jBPM, a BPMN-based BPM system developed by JBoss; a division of Red Hat, is one of widely used open source BPM systems (Harmon, 2007). This thesis tries to explore the capability of jBPM to facilitate performance measurement of care pathway implemented in it.

1.2 Research Objective

Based on the problem definition above, the objective of this research is:

Develop an approach to formalize performance indicators of care pathway that is executable on a BPMS

The approach in formalizing performance indicators must be:

- ✓ Comprehensive, i.e. it covers The Leuven Clinical Pathway Compass as much as possible.

- ✓ Expressive, i.e. it gives the users high level of freedom to define indicator formula.
- ✓ Generic, i.e. it covers as many types of care pathways as possible.
- ✓ Intuitive, i.e. it is easy for users to understand and implement.
- ✓ Executable against BPMS execution data.

The context of this research is illustrated in Figure 1.

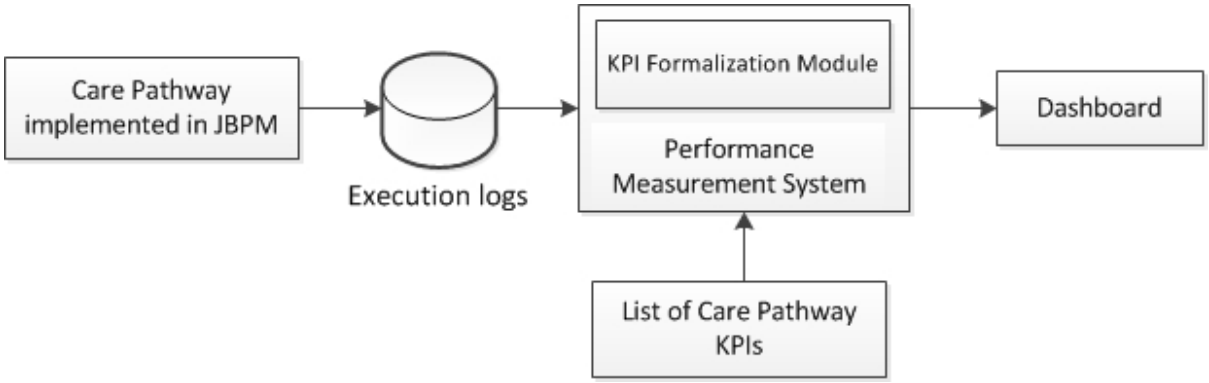


Figure 1 Context of the research

1.3 Research Question

The research objective above drives the following main research question:

Given a care pathway executable in a BPMS, i.e. jBPM, how can its performance indicators be formalized and calculated if The Leuven Clinical Pathway Compass is used as the guideline?

The main research question above can be divided into the following sub questions:

1. What is care pathway?
2. What is The Leuven Clinical Pathway Compass?
3. What is BPMS? What is jBPM?
4. What are the requirements to implement the formalization of care pathway performance indicators in jBPM?
5. How to design the formalization of care pathway performance indicators?
6. How to make the formalization executable?
7. How to evaluate the formalization?

1.4 Research Methodology

To satisfy the research objective and to answer the research question, the following research methodology is formulated:

1. Literature study on care pathway. It is important to understand the concept of CP as the problem domain of measurement and to understand how it is developed.
2. Literature study on CP performance measurement guideline, i.e. The Leuven Clinical Pathway Compass. The guideline will be the starting point to determine which dimensions or aspects are involved in measurement. To assess whether initiatives on formalization of care pathway performance indicator and on automatic performance measurement already exist, several related works are analyzed. Lessons learned from previous research can be taken into account in designing indicator formalization.

3. Exploration on jBPM. This step is necessary to understand its role in care pathway development and its capability in answering care pathway performance measurement needs, such as the availability of logs as the data source of measurement.
4. Analyze the requirements to formalize care pathways performance indicators. The analysis is solely based on literature study and does not involve domain expert. This is performed by analyzing the Leuven compass in detail and aligns it with the JBPM capabilities. Monitoring variances as a specific type of process monitoring in care pathway is also analyzed. From this analysis, a list of care pathway performance measures is produced. Note that the identified measures are generic for all care pathways implemented in jBPM.
5. Design indicator formalization.
6. Implement the formalization into programming code. A web interface representing performance measurement system will be developed to display the measurement scenario.
7. Evaluate the quality of formalization approach using selected software product quality metrics. To evaluate the functionality of the approach, a healthcare process representing care pathway, i.e. weaning process, is used. A list of KPIs of weaning protocol is gathered from a domain expert and is used for testing. The fulfillment of the criteria defined in the research objective and non-functional aspect of the approach are also evaluated.

1.5 Research Scope

A BPMN-based BPMS will be chosen to prove the concept. In this master project jBPM 6.1.0.CR will be used where jBPM web-based workbench is used to implement and execute the pathway. Initially, this project uses jBPM version 6.0.1 Final. In the middle of the research, JBoss releases a newer version of jBPM, 6.1.0.CR1, which accommodates better historical data logging and is considered useful for care pathway performance measurement purpose. Analysis of historical data logs is based on version 6.1.0.CR1 and the rest will be based on version 6.0.1 Final. The timeline of this change is shown in Figure 2.

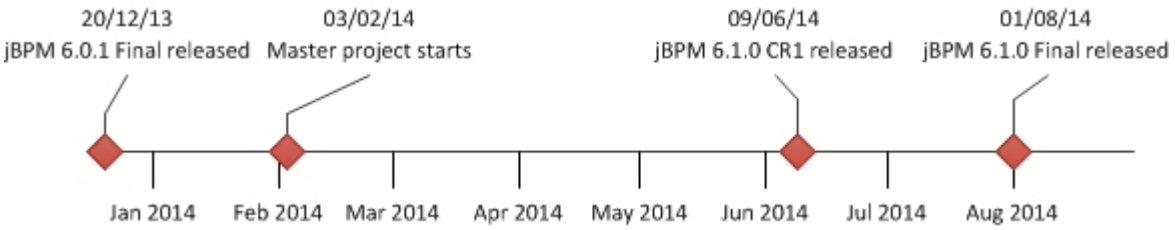


Figure 2 jBPM release dates

A performance measurement guideline, i.e. the Leuven clinical pathway compass, for care pathway will also be chosen as the basis to determine the scope of measurement. Consequently, the applicability of KPI formalization is limited to the chosen BPMS and guideline. To prove the concept, it is assumed that a care pathway modeled in BPMN, along with the definition of data elements and roles exists. The data elements covered in this project are of standard data types such as string, numeric, and Boolean.

1.6 Document Structure

The remainder of this thesis is structured as follows (corresponding research methodology for each chapter is illustrated in Figure 3).

Chapter 2 presents the result of literature study on care pathway, the Leuven compass, and exploration result of jBPM. The concepts in this literature study will be used throughout the thesis. Chapter 3 is dedicated to the analysis of the requirements to conduct performance measurement in CP implemented in jBPM. It will cover deeper analysis on the Leuven compass and jBPM. In addition, it will discuss variance monitoring as a special case of process monitoring in CP. The list of care pathway performance measures as the summary of requirement is also produced in this chapter. The design of the indicator formalization and the details of implementation of measurement system are presented in Chapter 4.

In Chapter 5, the result of system evaluation is discussed. Finally, Chapter 6 will conclude the research; discuss the limitations, directions for future works and discussion about related software technologies.

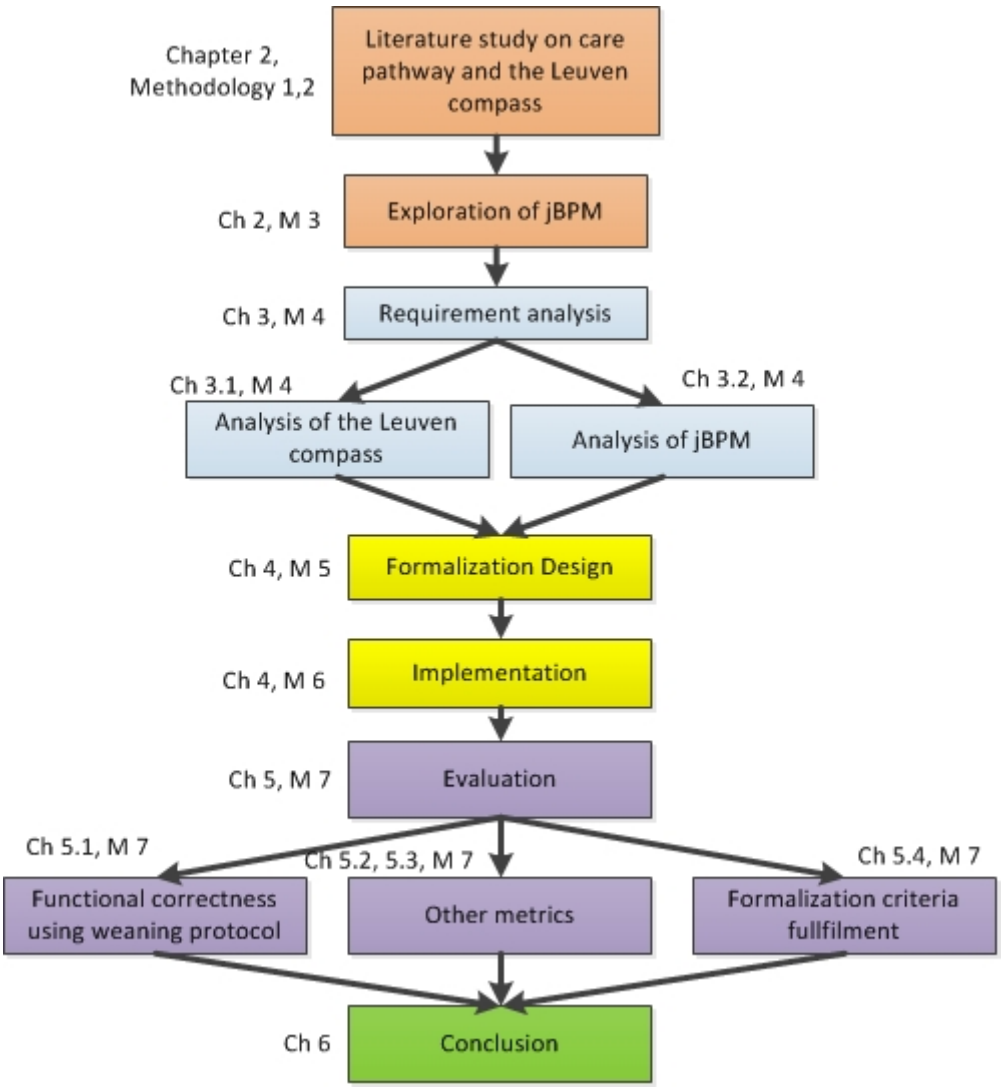


Figure 3 Thesis structure and research methodology

Chapter 2

Preliminaries

In this chapter, the result of literature study is presented. It will cover the discussion about challenges and opportunity in healthcare sector, care pathway as a solution, how care pathway is developed, and what guidelines in care pathway performance measurement exist. jBPM as an implementation environment of care pathway is also briefly discussed. A brief discussion about Structured Query Language (SQL) is also presented. Finally, this chapter is concluded by discussing several related works in care pathway performance measurement.

2.1 Healthcare Sector Challenges and Opportunity

Recent report shows that worldwide healthcare sector today is facing harder challenges than any time before. At the same time, it becomes opportunity for this sector to come up with new breakthroughs to tackle this issue. Deloitte reports that there are four factors leading to the challenges, which are aging population and chronic disease, cost and quality, access to care, and technology (Deloitte, 2014).

Considering cost and quality issue, healthcare providers are pressured to provide high quality of care while cutting the medical cost at the same time. Patients and policymakers, i.e. government, demand the increase to the access to care services and treatment advancements. On the other hand, the payers, e.g. insurers expect hospitals to lower the medical cost and promote evidence-based treatment. Unfortunately, a higher cost does not always correlate to higher quality of care. Sometimes, the problem does not lie in the disease of patients but at the hospital that treats them. Over prescribing of drug is one of such contradictory situation (Deloitte, 2014). In short, the problem might lie in the care process of hospitals.

The problems above push healthcare providers to perform their business process in more effective and efficient way without ignoring patient safety aspect. To achieve this goal, a lot of practical guidelines were developed in the past half century to assist physicians in making effective decision (Walker, Howard, Lambert, & Suchinsky, 1994). Another approach is by organizing care process, in which care pathway is one of methodologies (Vanhaecht, K.; De Witte, K.; Sermeus, W., 2007).

2.2 Care Pathway as a Solution

European Pathway Association (EPA) describes a care pathway (CP) as "*a complex intervention for the mutual decision making and organisation of care processes for a well-defined group of patients during a well-defined period*". The mutual decision implies that care pathways involve multiple professional groups (doctors, nurses, etc.) to interact in making decision (Vanhaecht, Panella, van Zelm, & Sermeus, 2010). In their website, EPA mentions several terminologies which are synonym to care pathways, i.e. clinical pathways, critical pathways, care paths,

integrated care pathways, case management plans, clinical care pathways or care maps (European Pathway Association, 2014). In the rest of this document, the term *care pathway* will be used.

EPA defines the characteristic of care pathway as follows:

"(i) An explicit statement of the goals and key elements of care based on evidence, best practice, and patients' expectations and their characteristics;
(ii) the facilitation of the communication among the team members and with patients and families;
(iii) the coordination of the care process by coordinating the roles and sequencing the activities of the multidisciplinary care team, patients and their relatives;
(iv) the documentation, monitoring, and evaluation of variances and outcomes; and
(v) the identification of the appropriate resources."

And the goal of care pathway is:
"… to enhance the quality of care across the continuum by improving risk-adjusted patient outcomes, promoting patient safety, increasing patient satisfaction, and optimizing the use of resources." (European Pathway Association, 2014)

Variance can be defined as any deviations from standardized pathway (Du, Jiang, Diao, Ye, & Yao, 2009) which is unexpected that can be positive or negative for patient outcomes (Hyett, Podosky, Santamaria, & Ham, 2007). It can occur for four reasons: patient condition, healthcare worker condition, hospital condition and society condition (Vanhaecht, K., & Sermeus, W., 2003). The variations are common in a high-risk type of business like healthcare.

The concept of care pathway was derived from industrial process in 1950s (Schrijvers, van Hoorn, & Huiskes, 2012) and was first introduced in the healthcare domain between 1985 and 1987 at the New England Medical Center in Boston (USA). It was introduced in Europe, i.e. UK in early 1990s and later disseminated worldwide in the late 1990s (Vanhaecht, Panella, van Zelm, & Sermeus, 2010).

In its early stage, the management of care pathways was implemented in a paper based system, i.e. by filling predefined forms (Du, G., Jiang, Z., & Diao, X., 2008). As the information technology advances, the electronic versions of paper-based care pathways have been also developed. In the early 1990, the linear sequential model of CP was designed. However, due to limitations of the linear model, the state-transition model was designed in the late 1990s. Finally, since the early of 21st century, there have been numerous proposals about structural design of the electronic care pathway (Wakamiya & Yamauchi, 2009). Nevertheless, the paper-based CP is still preferable by many healthcare providers due to high investment cost (Wakamiya & Yamauchi, 2006) despite its drawbacks such as limited capacity of data recording and collection, inability of forms in representing complex logical and timing relationship of different activities, and lack of support for monitoring and handling variations of CP based on knowledge (Du, G., Jiang, Z., & Diao, X., 2008).

The way care pathways being represented and developed is unclear (European Pathway Association, 2014). Several researches are conducted to propose the structural design and computerized implementation of care pathway. Ontology approaches are proposed to model information flowing in care pathways workflow (Ye, Jiang, Diao, Yang, & Du, 2009) (Daniyal,

Abidi, & Abidi, 2009) (Hu, et al., 2009). BPM approach is also proposed in several reports to model and implement care pathways as workflow applications. Du et al proposes implementation of CPs based on workflow in Petri net and a framework of CP adaptive workflow management system based on Extended Workflow-nets (Du, G., Jiang, Z., & Diao, X., 2008). Vermeulen in her master's thesis (Vermeulen, 2013) proposes a methodology to define care pathway as a process model with purpose primarily for communication tool between medical professional, technicians, and researchers. Renswouw in his master's thesis (van Renswouw, 2013) provides a guideline to translate a paper based care pathway into an executable workflow process model. He outlines several researches on care pathway implemented in workflow system and concludes that it is advantageous to transform existing paper based pathways into workflow applications. These works show that there is an increasing interest in adopting BPM discipline in care pathway domain and bring it into executable system.

2.3 Care Pathway Development

The development of care pathway can be seen as a continuous process of quality improvement. A 7-phase method was developed to design, implement and evaluate care pathways. It consists of screening phase, project management phase, diagnostic & objectification phase, development phase, implementation phase, evaluation phase, and continuous follow-up phase. This method is based on quality improvement method called Deming cycle, also known as "Plan-Do-Study-Act" cycle. Each phase in 7-phases method will pass a Deming cycle (Vanhaecht, et al., 2012). The 7-phase method is illustrated in Figure 4.

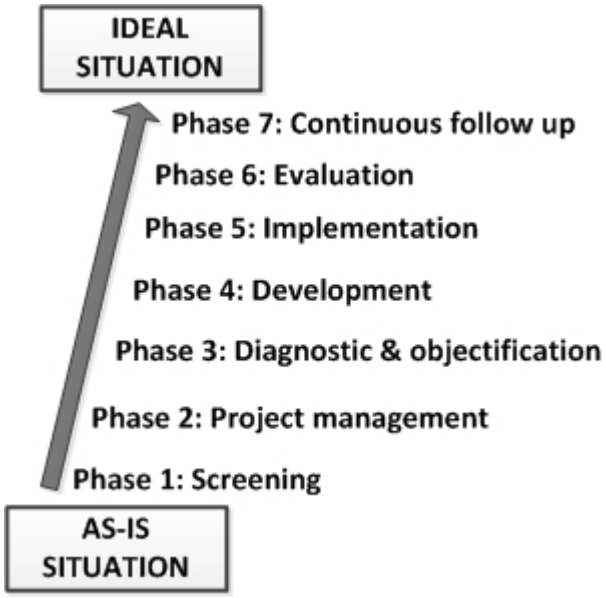


Figure 4 Seven phase method of pathway development (Vanhaecht, et al., 2012)

Screening phase can be initiated when there is a demand for a new pathway or there is a need to adapt or improve the existing one. It is performed by gathering and analyzing all information regarding the existing pathway or actual healthcare process.

After the decision to develop a pathway is clear, project management phase can be started. In this phase, the project related entities are defined, i.e. project team, goal, timeline, resources, and project charter. The care process where the pathway will be developed is also defined, including the patient group and the start and end point of the pathway.

In the third phase, the evaluation of as-is situation is conducted from four different perspectives: organization, patient, evidence and legislation, and external partners. This phase is considered very important as the objective information about current care process is resulted and will be the basis for evaluation phase.

Based on objective information collected in the previous step, the care pathway is developed in the fourth stage. The team may redefine the patient group, the start, and end time. The key

interventions are checked if they meet the process objectives, the sequence of activities and the corresponding actors are coordinated, and the required resources are identified.

After the pathway is developed, the implementation phase starts by drawing up an implementation plan and conducting an implementation test. The feedback from the test can be used to adjust the pathway before it is ready for daily practice.

In the evaluation phase, usability testing and examination of process and outcome indicators are performed. Continuous follow up aims to keep the pathway alive by monitoring evaluation results and making adjustment when necessary.

2.4 CP Performance Measurement Guideline: The Leuven Clinical Pathway Compass

The term performance is defined differently in many literatures. Lebas defines performance as "*the potential for future successful implementation of actions in order to reach the objectives and targets*" (Lebas, 1995). March and Sutton use the term *performance* and *effectiveness* interchangeably (March & Sutton, 1997). In this thesis, we stick to the definition of performance as "something accomplished" (Merriam Webster Dictionary, 2014). Thus care pathway performance measurement is defined as "the process of measuring what accomplished by care pathway implementation". This activity, which is the focus of this research, is basically related to the phase 6 and 7 in the seven-phase method of pathway development (Figure 4).

The definition of indicator is strictly linked to the concept of representation-target, which is the operation aimed to make a context (e.g. manufacturing process) or parts of it "tangible" in order to perform evaluations, make comparisons, formulate predictions, etc. (Franceschini, Galetto, & Maisano, 2007). Mainz outlines three definitions of (clinical) indicator as follows (Mainz, 2003):

1. As measures that assess a particular health care process or outcome.
2. As quantitative measures that can be used to monitor and evaluate the quality of important governance, management, clinical, and support functions that affect patient outcomes.
3. As measurement tools, screens, or tags that are used as guides to monitor, evaluate, and improve the quality of patient care, clinical support services, and organizational function that affect patient outcomes.

The essence of care pathways evaluation and continuous follow up is basically measuring indicators to assess the pathway performance. Based on literature, there are several conceptual tools exist for performance measurement in any business or in healthcare sector in particular, namely Balanced Scorecard, Clinical Value Compass and DataMap. However these tools are not fit for CP purpose. Balanced Scorecard and DataMap are suitable for strategic hospital level, but not suitable for operational level of CP. Clinical value compass is suitable for operational level, but it was initially developed for patient level, not group of patient level as required by care pathway (Vanhaecht, K., & Sermeus, W., 2003).

A tool called The Leuven Clinical Pathway Compass (see Figure 5) was designed as the guideline to evaluate and monitor the impact of care pathways implementation (Vanhaecht, K., & Sermeus, W., 2003). Instead of use at patient level, this conceptual tool is used at operational

level for a specific patients group. It contains five domains as the basis for the evaluation, i.e. clinical, services, team, process, and financial domain.

Indicators belong to clinical domain are directly linked to the disease and their impact. The specificity of disease implies that it is related to only specific group of patient as defined by the pathway. For Unstable Angina pathway, an example for this indicator is percentage of patients with Unstable Angina who are prescribed Aspirin at discharge (Meenakshy, 2013).

The service domain consists of indicators which measure the quality of service of the hospital or the team to the patient of particular pathway. Sample indicators belong to this domain are patient satisfaction, patient experiences and patient attitudes. The data for this indicator can be collected for example from questionnaire about the quality of care.

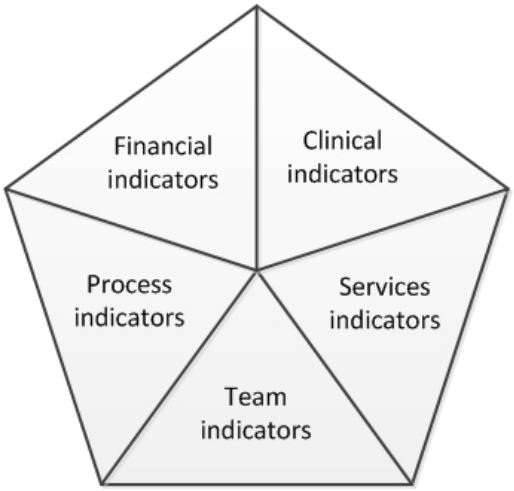


Figure 5 Leuven Clinical Pathway Compass (Vanhaecht, K., & Sermeus, W., 2003)

The third domain, team, comprises indicators that measure effectiveness of multidisciplinary team focusing on shared goals, clear role definition, clear procedures and good team relationship. Job satisfaction and coordination are also indicators belong to this domain.

The process domain covers indicators related to the "efficiency" goal of care pathways which monitors activities performed to give care. By measuring this type of indicators, it is expected that the bottleneck in the pathways is discovered and hence can be solved. Typical indicators belong to this domain are lead time, waiting times and time between two interventions. More specific clinical-time-related indicators can also be considered as process indicator. For example, percentage of patients with Unstable Angina who are handled within 15 minutes upon arrival is an indicator in Unstable Angina pathway. The monitoring or tracking of variances is also part of this domain (Vanhaecht, K., & Sermeus, W., 2003).

The cost related indicators are covered in financial domain. The measurement of this indicator is important to control the cost incurred in pathway while maintaining quality of care at the same time. This type of indicators can be derived, e.g. by creating the Bill of Services (BOS) for a pathway that lists all services like activities, intervention, goods, and assign the corresponding cost for each of them.

In this thesis, The Leuven Clinical Pathway Compass will be used as the basis for care pathway performance measurement.

2.5 Business Process Management System

There are many definitions of Business Process Management (BPM) in literature. In this thesis, we use one definition by Dumas et al which define BPM as "*a body of methods, techniques and tools to discover, analyze, redesign, execute and monitor business processes*". It can be seen as the art and science of controlling how work is performed in an organization with the goal to ensure consistent outcomes and to benefit from improvement opportunities. Thus a BPM system (BPMS) is defined as a software system which extends the functionality of traditional Workflow Management System, not only limited to work routing, but also on all activities of BPM. (Dumas, La Rosa, Mendling, & Reijers, 2013)

jBPM is an open source BPMS built on top of Java (distributed under Apache license). JBoss, a division of Red Hat, is responsible in developing the framework with support from community of developers (jBPM - JBoss Community, 2014) (jBPM Overview, 2014). We decide to use this BPMS in the project since it is one of widely used open source BPM systems (Harmon, 2007) and is continuously improved.

Until this master project was started, the stable final version released is 6.0.1. In the middle of this research, JBoss releases a newer non-final version of jBPM, 6.1.0.CR1, which accommodates better historical data logging capability. Observation over jBPM 6.0.1 shows that it does not implement human task audit capability that enables us to track when a human task is actually started and finished. In the latter version, it is implemented in package *org.jbpm.services.task.audit.impl.model.AuditTaskImpl* and a new table called *AuditTaskImpl* is added to the database. Version 6.0.1 also misses registration of the listener for task service which makes table *BAMTaskSummary* and table *TaskEvent* empty (JBoss Developer Forum, 2014). These tables are useful to analyze events related to task, e.g. when they are started and finished. It is fixed in version 6.1.0.CR1. Due to this change, the analysis of logs is based on the newer version. The analysis of the rest is based on the older version since it is already performed and there is no significant change in the jBPM core engine.

jBPM allows the users to model business process in BPMN 2.0. This process modeling language is widely adopted today as it provides a standardized graphical notation which is easy to use for business analysts to communicate with their partners (Volzer, 2010). Object Management Group (OMG), the organization that defines the standard for BPMN, reports that there are 74 vendors that implement BPMN 2.0 in their tools (Object Management Group, 2014). This version of BPMN added a standardized execution semantics which allows vendors to implement interoperable execution engines (Volzer, 2010). However, jBPM only supports (significant) subset of BPMN 2.0 elements and attributes, i.e. it only focuses on (almost) all executable parts (jBPM Processes, 2014). A complete subset of BPMN 2.0 that is supported by jBPM can be found in APPENDIX A.

2.5.1 jBPM Components

The components of jBPM project are explained as follows (jBPM Overview, 2014) (see Figure 6):

Execution module

The core engine is the most important part of the project that manages the execution of business process. Human task service is optional core that will handle human task life cycle if the process involves human actors. Persistence module is also optional if the users want to persist the state of process instances and log historical information of process execution in a database instead of keeping it in the system memory. By default, jBPM installation uses H2 database, which is an in-memory database that relies on the availability of main memory. H2 is still under testing and hence the reliability is questionable. Some users report that after a power failure, the database cannot be opened (H2 Database, 2014). Thus it is better to use disk-storage-based database system such as MySQL, PostgreSQL or Oracle. Persisting historical data is useful if the users want to perform further analysis on history information, for example performance measurement.

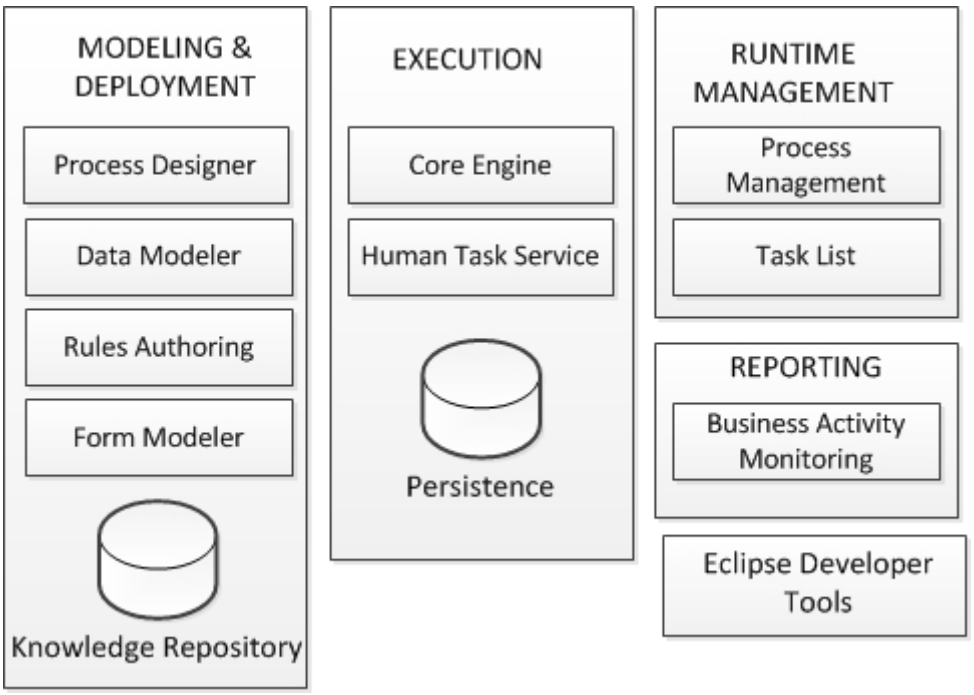


Figure 6 jBPM 6 project components (adapted from (jBPM Overview, 2014))

Modeling & Deployment module

Modeling and deployment of process can be done in a web-based environment. Process designer is responsible in providing the users with interface to graphically model processes in BPMN 2.0. This graphical model is stored as an XML file which is compliant to BPMN 2.0 specifications. Data modeler is used to view, modify, and create (non-standard, custom) data model that is used in the processes. Rules authoring module enables the users to define business rules that are used along with the business process. Form modeler allows the users to create, generate or modify forms related to process. The artifacts produced in modeling and deployment part is stored in a repository called Knowledge Repository.

Runtime management module
jBPM also provides the users with a web-based console to manage the process like start new process or abort existing process instance, and to manage the task list.

Reporting module
Reporting module, called Dashboard builder or simply Dashbuilder, provides Business Activity Monitoring (BAM) capability to jBPM. There are two types of dashboard provided by Dashbuilder: business dashboard and process dashboard.

Business dashboard module allows users to develop dashboard in which the data can be retrieved from both relational databases using SQL query and CSV files. The data for this dashboard are not necessarily from the database being used by jBPM (i.e. if persistence option is enabled) but can be from any relational databases. Process dashboard is a specific version of business dashboard, where the data source being used is jBPM database, i.e. from table *ProcessInstanceLog* and *BAMTaskSummary* (see Figure 7 and Figure 8).

Eclipse developer tools
Eclipse developer tools are a set of plugin to the Eclipse IDE which enable developers to integrate business process in the development environment.

All components above (except Eclipse developer tool) are combined into a web-based application called *KIE workbench*. KIE refers to the project in which jBPM belongs to. The workbench facilitates complete cycle of process, from modeling, implementation, execution and monitoring. The user can also perform simulation from workbench. However the history information will not be logged by automatic simulation.

2.5.2 jBPM History Logs

As explained before, jBPM allows its users to persist data related to process execution in a relational database management system such as MySQL, PostgreSQL, Oracle DB and Microsoft SQL Server through its persistence module. The log is called history log, also known as audit log, and consists of three tables: *ProcessInstanceLog*, *NodeInstanceLog*, and *VariableInstanceLog* (see Figure 7). This master project uses PostgreSQL to store the history log. The steps performed to make jBPM be able to persist data in PostgreSQL are explained in APPENDIX B.

Table *ProcessInstanceLog* stores basic information about process instance, for example the start date and the end date. Table *NodeInstanceLog* stores information about which nodes in the process model, for example tasks, were truly executed in a process instance. Table *VariableInstanceLog* stores data about the changes in process variable instances of a process instance. Process variables can be defined as data elements that are specified for a process and can be modified by user (or task) when executing process tasks. The complete explanation of the definition for each column of tables above can be found in APPENDIX C.

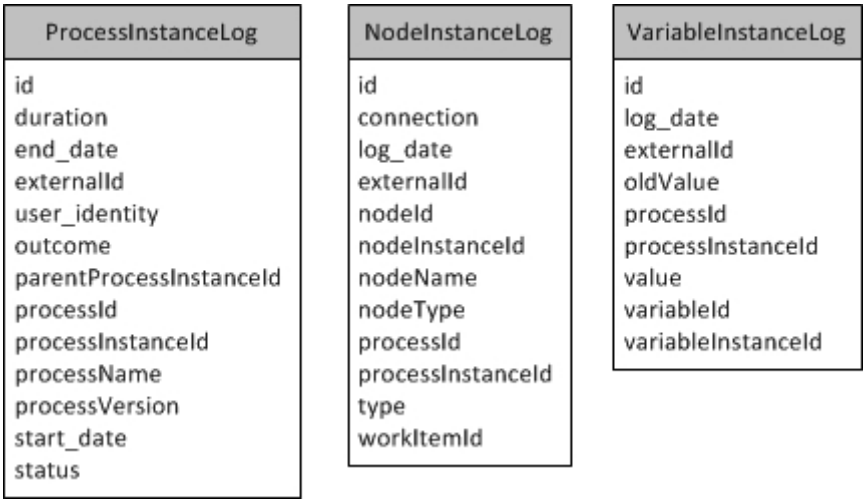


Figure 7 jBPM history log tables

In addition to the tables above, jBPM also stores information about task in three other tables, namely *BAMTaskSummary*, *AuditTaskImpl*, and *TaskEvent* (see Figure 8).

Table *BAMTaskSummary* contains summarized information about tasks for all process instances which basically stores creation date, start date and end date of the tasks. It only stores the last state of tasks. This table is summarized from table *AuditTaskImpl* and *TaskEvent*. Table *AuditTaskImpl* contains unique records of tasks created. The detailed task status changes are described in table *TaskEvent*.

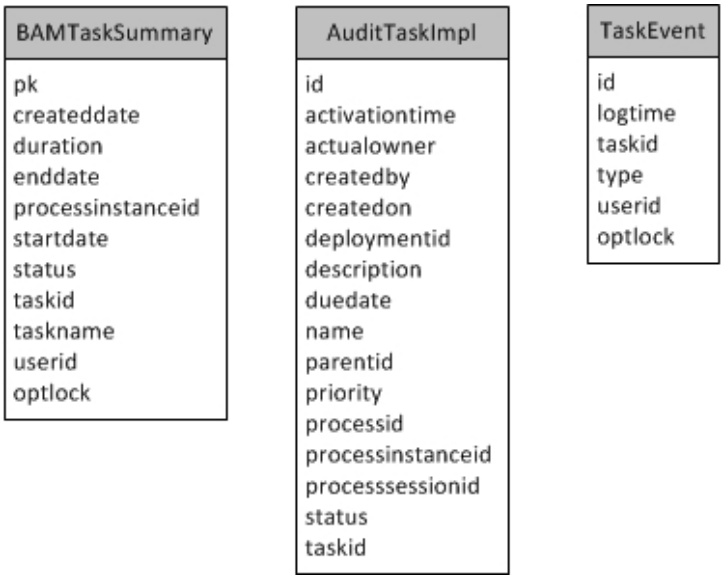


Figure 8 jBPM task-related tables

2.6 Structured Query Language (SQL)

Since jBPM can store its history logs in relational database system, it is necessary to understand SQL as the language to access them. SQL is a set-oriented programming language that is used to interact with relational database systems. Relational database stores information in tables which is divided into rows and column. SQL allows the users to create tables, insert rows into

tables, update data, delete rows and query the rows in tables (IBM, 2014). A typical of SQL query has the form as shown in Figure 9 (Silberschatz, Korth, & Sudarshan, 2011).

```
select A1, A2, …, An
from r1,r2, …, rm
where P;
```

Figure 9 Typical SQL query format

Each A_i represents an attribute (for example a column) *and each r_i a relation* (for example a table). *P is a predicate*. *If the **where** clause is omitted, the predicate P is **true**.* (Silberschatz, Korth, & Sudarshan, 2011). Predicate can be considered as filter or constraint that should be satisfied by returned records.

The calculation of performance indicators is basically passing SELECT query to the data source, by specifying relation(s) and predicate and applying post query tasks to the returned record, e.g. applying aggregate function.

In spite of its strength in data retrieval, SQL has some weaknesses. It can facilitate the users to retrieve information hidden somewhere in a single-table or multitable database without the need to know the order of the table's rows or column. SQL can also extract the data we want regardless the number of rows in the table by single SELECT statement. On the other hand, SQL cannot easily operate on one row at a time. Sometimes we want to perform specific operation for each row individually (Taylor, 2010) or we want to perform complex operation involving multiple query results.

In the case above, we can make use of the capability of procedural languages, which is designed for one-row-at-a-time operations and is able to accommodate complex operations. However, data type compatibility must be considered if we want to combine SQL with procedural language (Taylor, 2010).

2.7 Related Works

As stated in chapter 2.2, the way care pathways being developed is unclear; hence many attempts are performed to implement it. Consequently, the way performance being measured is different for different enactment approach. A research revealed that there is a lack of continuous follow-up after the care pathway implementation. Information and communication technology (ICT) support is also one of main challenges in care pathway implementation (van Gerven, Vanhaecht, Deneckere, Vleugels, & Sermeus, 2010). To see whether initiatives on formalization of performance indicator; especially in care pathway and its implementation using ICT support exist, several works are investigated.

Several studies try to evaluate and monitor pathway implementation. Aledo et al report the result of evaluation and monitoring of a clinical pathway for thyroidectomy one year after the implementation and after four years' follow up in a hospital in Spain (Aledo, et al., 2008). Similar research was also performed for colorectal cancer in 2011 which evaluate the pathway implementation over a 5-year period (Aledo, et al., 2011). Regardless the result of the evaluation, by looking at the period of monitoring from those two researches, it is concluded that continuous follow up is rarely performed and ad-hoc performance measurement is performed due to the unavailability of automatic performance measurement system.

A research tries to measure care pathway performance using semantic web technology approach (Meenakshy, 2013). In her master's thesis, Meenakshy uses ontology to formalize indicator and translates it into SPARQL query to retrieve the data from RDF database. The RDF database itself is derived from extraction of electronic patient record (EPR), which is a relational database. However, the way the data is retrieved, i.e. the selection of tables as data source, from EPR is not clearly explained. Moreover, different pathway might have different definition of ontology which makes extensive work must be performed to construct the ontology.

An approach in a more generic business setting was proposed by Caputo et al with the aim to define, formalize and model KPI according to Model Driven Architecture vision and to integrate the definition of KPI into the business model. It suggests the use of Balanced Scorecard and the Goal Question Metric to identify KPIs and realize process model through BPMN. In the approach, KPIs are expressed as formula and formalized in XML format. The formula combines performance measures to produce performance indicator. It is outlined in the work that measure is distinguished from indicator. "*A 'Performance Measure' is a description of something that can be directly measured (e.g. number of reworks per day). A 'Performance Indicator' is a description of something that is calculated from performance measures (e.g. percentage reworks per day per direct employee).*" (Caputo, Corallo, Damiani, & Passiante, 2010). However, the approach does not explain how to link the formalization with process model and more importantly, to make it executable.

The related works above show that the proposed techniques in care pathway performance measurement, especially in performance indicator formalization are subject to improvement. Based on the gathered knowledge about care pathway and implementation environment of care pathway (i.e. jBPM), a new technique in care pathway performance measurement is proposed. The requirement to develop this technique is discussed in the following chapter, Requirements Analysis.

Chapter 3

Requirements Analysis

In this chapter, the requirements to perform performance measurement for care pathways are analyzed. The purpose of this analysis is to match the measurement scenarios as described by CP compass with the capability of jBPM in providing required information. The analysis is concluded by identifying care pathways performance measures. This analysis will be the basis for the design of KPI formalization in the next chapter. The aspects covered in this chapter are: analysis of The Leuven Clinical Pathway Compass, analysis of jBPM, monitoring variances, and concluded with identifying CP performance measures.

3.1 Analysis of the Leuven Clinical Pathway Compass

The Leuven Clinical Pathway Compass defines five domains for CP performance measurement. However, in practice, not all indicators in all domains should be measured for each pathway (Vanhaecht, K., & Sermeus, W., 2003). It depends on the requirements of the team in charge of the pathway to define the indicators. Nevertheless, this project will analyze all domains to understand the requirement for each of them for performance measurement purpose.

3.1.1 Clinical Indicators

This type of indicator is based on standards of care which can be evidence-based or derived from the academic literatures. If scientific evidence is not sufficient, an expert panel of health professionals will determine it in a consensus process based on their experiences (Mainz, 2003). Regarding quality improvement purpose, clinical indicators can be classified into three categories, i.e. structural indicators, process indicators, and outcome indicators. Structure refers to the attributes of the settings in which the care is provided, i.e. material resources, human resources, and organizational structure. Process refers to what is actually performed in giving and receiving care. Outcome denotes the impact of care on the health status of patients or population (Donabedian, 1988), e.g. blood pressure results for hypertensive patients and mortality rate (Mainz, 2003).

Although Donabedian's classification above is widely used today (Aboriginal Health & Medical Research Council, 2013), in order to avoid confusion with clinical indicators defined in CP compass, structural indicators and process indicators above are mapped as process indicators in CP compass. Thus outcome indicators are considered as the only clinical indicators.

There are two types of (clinical) indicators (Decker, 1991):

1. *Rate-based indicator*. This type of indicator is expressed as proportion or rates, ratio, or mean values for sample population (Mainz, 2003). The evaluation of this type should define

the time interval of surveillance, collecting the number of occurrence of events of interest during that interval (i.e. the numerator), and appropriate denominator. Denominator is usually the total number of events where patients are at risk of appearing at the numerator in the same interval. For example, *percentage of patients who got infection during hospital stay from February 2014 to May 2014* is a rate-based indicator with the total number of patients who got infection in that period is the numerator and the total number of patients in that period is the denominator (Meenakshy, 2013).

2. *Sentinel indicator*. This indicator monitors occurrence that should happen all the time or should never occur. It identifies individual events instead of ratio, and hence further investigation is conducted in case such events occur. For example, the number of patients who die during surgery.

3.1.2 Process Indicators

As stated in clinical indicators section above, process and structure indicators in Donabedian's classification are mapped as process indicators in CP compass. Hornix, in his master's thesis, outlines several process-related KPIs as follows (Hornix, 2007):

1. **Throughput time** (also known as lead time), the total time that the case spends in the process. It is formulated as time difference between the end event and the start event. In pathway context, it can be considered as the length of hospital stay and can be formulated as time difference between hospital discharge and patient arrival.
2. **Arrival rate of cases**, is the number of cases that arrive to the process per time unit. In pathway context, it is equal to the arrival rate of patients.
3. **Number of cases**, equals to the number of patient (in a specific time interval).
4. **Number of fitting cases**, is the number of cases that fit to the process model. Since the execution of pathway is based on the process model, the log will always conform the model. Thus this KPI can be omitted in the analysis.

Hornix also mentions several activity-related KPIs as follows:

1. **Waiting time of an activity**, is defined as the time between the schedule and the start event of that activity.
2. **Execution time of an activity**, is defined as the time between the start and the complete event of an activity without suspend time.
3. **Sojourn time**, is defined as the time between the schedule and the complete event of an activity.
4. **Arrival rate of activity**, is defined as the number of times an activity is scheduled per time unit.

Not all logs will store information about schedule, start, complete, resume, suspend time (Hornix, 2007). Some logs might only store start and end time. Consequently, not all activity related KPI above can be calculated.

Another important metrics in care pathway is time between two interventions, in this case two activities in the pathway workflow. It is defined as time difference between the starting time of two activities.

From the list of KPI above, it is concluded that:

1. The execution of pathway should store information about the time of interventions in its history log. Depend on the availability of time information in the logs, some KPI will be chosen.
2. Conditions are also specified to filter the collection of objects of interest, for instance time interval.
3. Arrival rate is a composite indicator in the form of proportion, where the numerator is the number of arriving case and the denominator is the time interval.

3.1.3 Financial Indicators

This indicator can be derived by developing Bill-of-Services of a pathway. It lists the cost for each intervention and price of resource required in a pathway. The measurement of this indicator should consider the following aspects:

1. **The number of execution of particular intervention**. This is important to consider it since some interventions might be performed multiple times, for example blood test. Multiple actions mean more cost. In workflow context, an intervention is implemented as a task, thus it is important to measure the number of execution of a task. This number will be the multiplier for the cost of intervention. An example indicator related to the number of execution is the total number of x-ray test for each patient in carcinoma pathway in the last three month.

2. **The quantity of resource being used**. Resource can be human resource, i.e. medical persons, or consumable resource, e.g. medicine or syringe. In workflow context, resource can be implemented in some ways. Human resources can directly act as actors of tasks. Other type of resource such as consumables can be implemented as a data element of process. In this thesis, we focus only on consumables medical resources.

3. **The cost for each intervention and resource**. In order to be able to calculate the real cost of pathway execution, it is necessary to store the information about cost for each intervention and resource.

3.1.4 Service Indicators

This indicator can be measured using an instrument called the Patient Perceived Quality of Care Questionnaire. Belgian-Dutch CP Network translated and remodeled it into 20-questions survey. (Vanhaecht, K., & Sermeus, W., 2003). This indicator is relevant to implement in the context of process model since a pathway process instance corresponds to a patient. Thus an activity in pathway model can be used to collect the data. For example, the collection of the data can be done during patient discharge. However, using multiple questions survey, a further analysis is required to understand the result. To simplify the measurement of this domain, a single question survey to rate the care service with response categories ranging from 1(worst) to

10(best) can also be used. In workflow context, it is implemented as a data element representing the rating.

3.1.5 Team Indicators

According to Vanhaecht in (Vanhaecht, K., & Sermeus, W., 2003), team indicator can be measured by means of survey for the medical team. He and his network developed the Leuven Team Effectiveness Scale for this purpose (Haspeslagh, et al., 2002). This instrument is basically a questionnaire consisting 22 questions. First 15 questions are about goals, roles, procedures and interpersonal relationships (GRPI model) with response categories ranging from almost never to almost always. The remaining 7 questions ask the same model with response categories ranging from one team to the entire team.

However, since it requires a deep analysis method for the questionnaire and is less relevant in the context of care pathway process model, we exclude this indicator from further implementation. The irrelevance exists since the flowing object in pathway process is the patients and the survey is not conducted for each pathway case but after some period of implementation.

From the analysis of the Leuven compass above, we argue that the classification of indicators by Decker is oversimplified, which covers only number of occurrence of event of interest. It cannot classify a measure, for example aggregate value such as minimum or maximum value. Hence we classify performance indicator (not only in clinical domain) as follows.

Value. This type of indicator considers the value of an attribute of single event of interest or a property of a measurement. Attribute value represents outcome of a care treatment and can be expressed as a value of a data element in workflow context. An example for an attribute value is the blood pressure of a patient in a collection of Unstable Angina patient record. An example for property of a measurement is time interval of measurement, e.g. one year from 1 January 2014 until 31 December 2014.

Aggregate. This type of indicator considers value of an attribute from multiple events of interest by applying an aggregation function to produce single scalar value, such as summation, means, median, mode, counting the number of elements, getting the minimum value, getting the maximum value, standard deviation, and variance. An example indicator belongs to this type is the average body temperature of Unstable Angina patient.

Composite. This type of indicator contains one or more elements, i.e. any combinations of value and aggregate indicators. The combination uses arithmetic operation such as division, multiplication, addition, etc. Rate and proportion/ratio/percentage are some kinds of indicator belong to this type. For instance, arrival rate of patient is the combination of number of patient (aggregate of patient) and time unit (value indicator). It is formulated as number of patient per time unit, i.e. using division operation. Another example is percentage of patient who is prescribed aspirin during hospital stay. It combines the number of patient who is prescribed aspirin (aggregate), divided by total number of patient (aggregate).

In relation with the differentiation between measure and indicator (see Chapter 2.7), value and aggregate indicators can be considered as measure and composite indicators can be considered as indicator. In this thesis, the term “indicator” refers to composite indicator. Therefore, it is

necessary to identify what elements that compose the (composite) indicators with respect to four domains in the Leuven compass, i.e. what measures in which their collection of values are aggregated.

In addition, there are conditions that should be met when collecting the event of interest. For example, time interval of data collection. In a more specific case, for example, only return patient with temperature 37'C. Condition is defined as the requirements that should be satisfied by event of interest.

To conclude this section, the proposed solution will only support the measurement of clinical, process, financial, and limited support for service indicators.

3.2 Analysis of jBPM

This analysis is performed to see the availability and the structure of information provided by jBPM to support care pathway performance measurement scenarios as implied in the analysis of the Leuven compass. It will cover the setup for jBPM, the scope of the analysis, the analysis of jBPM history logs and BPMN elements, and the analysis of supported data types. In workflow context, a care pathway can be associated with a workflow, in which the start event and the end event are defined.

3.2.1 jBPM Setup

jBPM version 6.1.0.CR is used as the implementation environment of care pathway. Before performing the analysis of jBPM, the following settings are applied:

1. KIE workbench is used as the implementation environment of care pathway. Although we can develop our own pathway application and embed jBPM core into it, this project will use existing application, i.e. KIE workbench, with its default implementation setting (except persistence) to run the pathway.
2. Persistence feature is activated to log execution data. It is important to persist the execution data since it will be used as the data source for performance measurement. The log data will be stored in PostgreSQL.

3.2.2 Scope of Analysis

jBPM implements a lot of BPMN elements in its process modeling module. However, due to time limitation, this master project only analyzes some elements that are considered influencing or related to the history logs. Analysis of the logs is important to be able to formulate indicator queries correctly. The analysis will cover the following aspects:

1. Relationship among history logs.
   This analysis is important to see how the log tables are related each other and hence join operation can be correctly formulated.
2. *ProcessInstanceLog* table.
   Analysis of this table is required to see what process instance's properties can be used for performance measurement purpose, for example, process instance state, start time, and end time.

3. *VariableInstanceLog* table
   This table stores data elements/process variables that are manipulated during process executions. The analysis is necessary to understand how the variables are stored and structured and how can it be used to measure performance.
4. *NodeInstanceLog* table
   This table stores information about which nodes in the process model are actually executed. The analysis is performed to see how the nodes are stored and identify what information can be derived from the log.
5. *BAMTaskSummary, AuditTaskImpl,* and *TaskEvent* table
   These tables contain information about task performed in the process instances execution. The analysis is important to identify what information can be used for performance measurement purpose.
6. BPMN elements
   Some BPMN elements are chosen and analyzed to see its effect on the history log. The selection considers elements that appear in the resulting BPMN model in (Vermeulen, 2013) and (van Renswouw, 2013).
7. Temporal perspective
   This perspective is used in many indicators as shown in the analysis in chapter 3.1. Thus it is important to understand which time information is stored by jBPM.

3.2.3 Analysis of jBPM History Logs and BPMN Elements

The structure of jBPM logs is not explained in detail in the documentation. For information that is not explicitly explained in the documentation, experiments and observations are conducted in the KIE workbench and PostgreSQL administration consol. It is also possible to browse through all the source codes to see how jBPM works. However, due to time limitation, the code browsing is limited. For each aspect mentioned in chapter 3.2.2, the analysis is explained as follows.

Tables Relationship

The logs of jBPM are stored in a relational database. However, the relationship (e.g. foreign keys) among log tables is not explicitly defined in the database. Even though the field names might imply the relationship (e.g. column *processinstanceid*), a test using Scenario 1 is performed to prove this pre-assumption.

Scenario 1 Obtaining tables relationship

A model with single user task called task1 and a variable called var1 is used in the test (see Figure 30 in Appendix D.1). In task1, the user can modify variable var1. The resulting log data from a complete execution of the process for each table is given in Figure 33 in Appendix D.4.

By observing the log data and the definition of log tables, it is concluded that the relationship among log tables is depicted in Figure 10. Note that this is not the definition of *foreign key* since column *processinstanceid* and *taskid* are not primary key in table *ProcessInstanceLog* and *AuditTaskImpl* respectively. Several experiments show that the value of those columns is unique and always equal to the value of column *id* in both tables. Thus it is concluded that referencing using column *processinstanceid* is not a problem.

ProcessInstanceLog table

Some fields in this table require additional information and are explained as follows.

1. *Processinstanceid* can be associated with patient id. Thus single process instance represents a patient.
2. Column *duration* stores actual duration of process instance since its start date in millisecond. The formula is *end_date – start_date*. For uncompleted process instance, the value is NULL since *end_date* is not known yet. The value represents length of hospital stay.
3. Column *externalId* stores external identifier (optional) e.g. deployment id of a process business. In jBPM it is possible to assign a version number for a process business deployment. Consequently, similar process (with similar *processid*) might have different *externalId* and *processversion*. In this project, *processid* is considered as the identifier of the process. In KIE workbench, information about *processid* of a process can be found in menu Process Management → Process Definitions → Double click on a process name to see its details → Definition id is the processid.

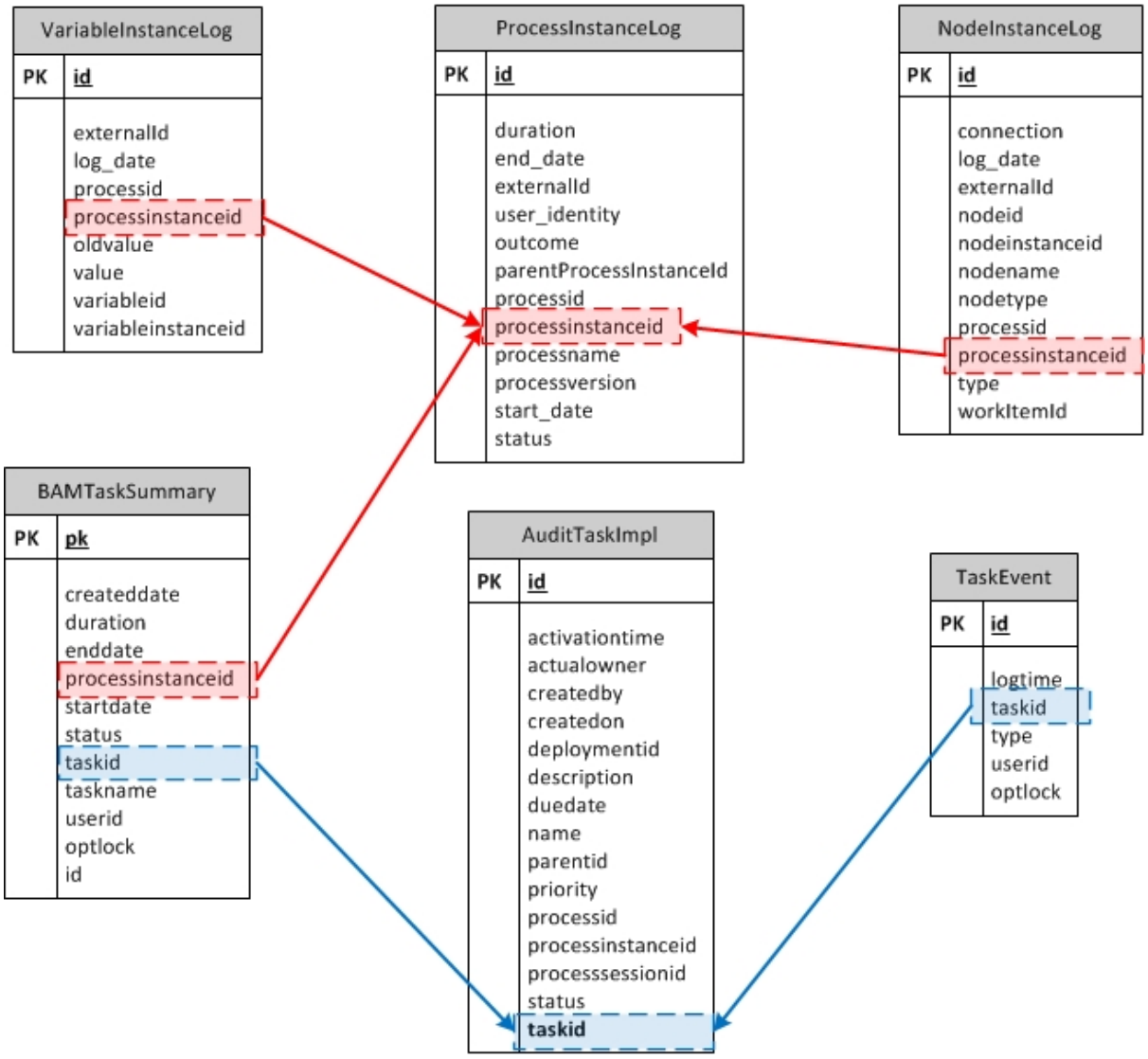


Figure 10 Log tables relationship

4. Column *outcome* will store the outcome of the process instance, e.g. error code. This error code can be specified in property *ErrorRef* of Error End Event.
5. The instances of reusable subprocess are treated as a separate process instance, and hence column *parentProcessInstanceId* will store *processinstanceid* of main process which uses that subprocess.
6. Column status stores information about status of a process instance. jBPM defines 5 states for process instance represented as integer value as described in Table 1 (Constant Field Values, 2014) (Interface ProcessInstance, 2014). Every time the status of process instance is changed, the value of field status in table *ProcessInstanceLog* for corresponding process instance will be updated.

Table 1 Process instance states

State	Value
STATE_PENDING	0
STATE_ACTIVE	1
STATE_COMPLETED	2
STATE_ABORTED	3
STATE_SUSPENDED	4

From the observation of KIE workbench, it does not provide any means to change the state of process instances to STATE_PENDING and STATE_SUSPENDED. Therefore, the analysis is limited to the three other states. STATE_ACTIVE means that the process instance is created but it has not reached the *End Event* (in pathway context: a patient is registered in the pathway and is in the middle of treatment). STATE_COMPLETED means that process instance has reached the *End Event* for any reasons, such as completion through happy path or completion through other end event types like error event (in pathway context: a patient has already discharged from pathway for any reasons, e.g. finished treatment or transferred to other protocol). STATE_ABORTED means that the process instance is aborted when it is in state STATE_ACTIVE (in pathway context: the treatment process is discontinued in the middle of process, for example: the patient decides to leave the hospital before the treatment process complete).

VariableInstanceLog table

By observing the execution result (Figure 33, Appendix D.4) in table V*ariableInstanceLog* of Scenario 1, it is concluded that each row stores a variable and its corresponding value. Regardless the variable data type, the value is stored in string data type in the database. An important thing to consider is to observe the impact of variable value change to the record, whether a new record with new variable value is inserted or the same record is preserved with value replacement approach is used. To perform this we use Scenario 2.

Scenario 2 Checking the impact of variable value change to table *VariableInstanceLog*

A test is performed using a process model containing two tasks in a sequence (see Figure 31 in Appendix D.2). In task1 and task2, the user can modify a process variable var1. The result is shown in Figure 34 in Appendix D.4.

In conclusion, all variable changes are stored. Hence the possible variable values derived from variable changes are: the latest value, the number of value changes and the initial value. In case the data type is numeric, we can derive the minimum value, the maximum value, the sum of values and the average value.

***NodeInstanceLog* table**

This table logs which nodes were actually executed during process instance runtime. The log is inserted whenever a node instance is entered from one of its incoming connection (type=0) or is exited through one of its outgoing connections (type=1).

By observing execution result in table *NodeInstanceLog* of Scenario 1(Figure 33, Appendix D.4), it is concluded that:

1. The identifier of node is column *nodeid*. This information is not directly visible from graphical representation of process model and can be obtained from the XML file of the corresponding BPMN process model (property *id* of elements under <bpmn2:process>).
2. jBPM does not store the time when a task is actually executed in this table. However, it stores the time when a task is ready to be executed which is the scheduled time of the task. It is derived from the *log_date* of the task where type=0.
3. Sojourn time of a task can be derived from the difference between *log_date* of entering record and exiting record of the task.

***BAMTaskSummary, AuditTaskImpl,* and *TaskEvent* table**

These tables store information related to human tasks performed during process instance execution. In table *BAMTaskSummary* we can see when a task is created (i.e. activated, which implies scheduled time), start time and end time. This table is summarized from table *AuditTaskImpl* and *TaskEvent* which store task-related information in more detail, recording all states a task has passed.

An experiment is performed to see what effects are produced in table *BAMTaskSummary* if a task does not follow normal lifecycle, i.e. created → started → completed, for example if a task is suspended. First, we will see what states a task might have. jBPM implements WS-HumanTask specification for its task lifecycle (jBPM Human Tasks, 2014). It is illustrated in Figure 11.

A normal flow is explained as follow. When a task is newly created, it is in *Created* state and then immediately becomes *Ready*. In this *Ready* state, the task will appear in the actors' task list page until an actor claims it. After the task is claimed, it will change to *Reserved*. If a task was created in which only one actor is potential to do the task, it will immediately change from *Created* to *Reserved*. The status changes to *InProgress* when the actor starts executing it. Once the actor has performed and completed the task, it will change to Completed. All status changes are recorded in table *TaskEvent*.

Sometimes a task is released by the actor in the middle of task execution that makes a task is suspended. To see its impact to the log, the Scenario 3 is tested.

Scenario 3 Observing the impact of releasing a task to the log

A process model (Figure 30, Appendix D.1) is used where the user can modify a process variable var1. The following task execution scenario is applied: start task → release task → claim task → start task → complete task. The result in the log is shown in Figure 35 in Appendix D.4.

By observing the test result, it is concluded that:

1. State ADDED is the state where a task is activated. The time when it is logged implies the schedule time of the task.
2. State STARTED is the state where the user actually starts the task. The time when it is logged is the starting time of task.
3. In case task is interrupted in the middle of execution (e.g. released), a new record with status STARTED is added after the task is re-started. Hence the actual execution time is derived from time difference between task completion (status COMPLETED) and the last start time.
4. Sojourn time is defined as time difference between status ADDED and status COMPLETED.
5. Waiting time is defined as time difference between status ADDED and the last status STARTED.

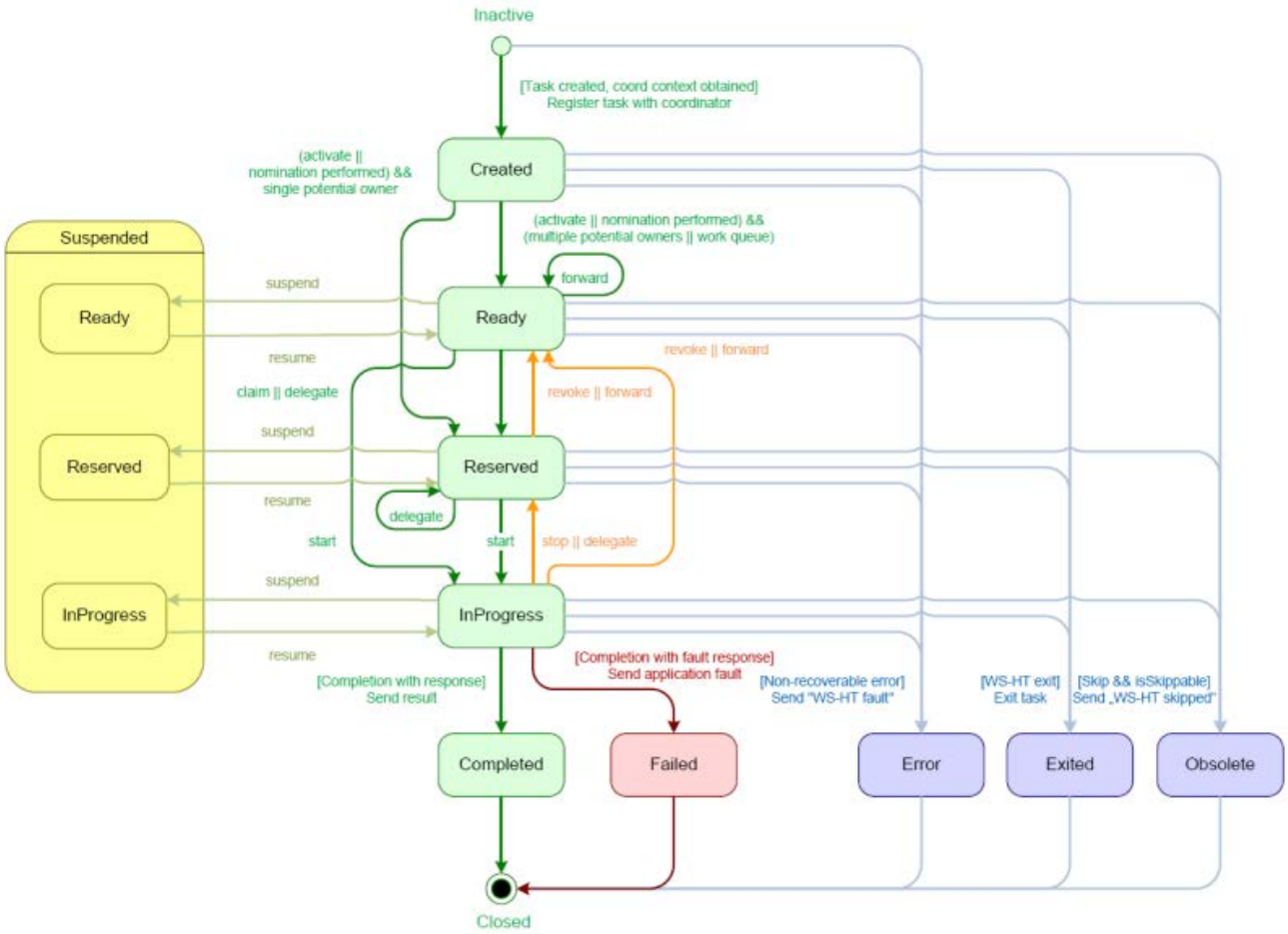


Figure 11 WS-HumanTask lifecycle

BPMN elements

The analysis is necessary to see what impacts can be concluded from implementing BPMN elements to the history logs. In this analysis, only elements that are found in pathway models (van Renswouw, 2013) (Vermeulen, 2013) and need additional explanation are covered. The analysis of exception handling will be discussed in chapter 3.3 about monitoring variances.

1 Task

jBPM supports many types of task: User, Send, Receive, Manual, Service, Business Rule and Script. Although they have different implementation in the background, as long as the task modifies process variables, it will be recorded in the table *variableinstancelog* with no differentiation in term of structure. Execution of any task will also be logged in *nodeinstancelog*. In conclusion, regarding log records, there is no specific behavior can be deducted from tasks.

2 Subprocess

The analysis is limited only to Reusable Subprocess. This type of subprocess will invoke another process (called child process) to be used within the main process. The important properties to be set are:

Called element, the id of the process to be invoked.

Wait for completion. If it is set to true, the parent process will continue only after the child process completes its execution, either aborted or completed. Otherwise, the parent process will immediately continue. By default it is set to true.

Independent. If it is set to true, the child process will not be terminated if the parent process is completed. It can only be set to false only when *Wait for completion* is set to true. (jBPM Processes, 2014).

To see the impact of implementing Reusable Subprocess to history logs, the Scenario 4 is tested.

Scenario 4 Observing the impact of implementing reusable subprocess to execution log

A process model containing subprocess is used (Figure 32, Appendix D.3). In Subprocess sub1, the property Called element is set to process model as shown in Figure 30 in Appendix D.1. Both Wait for completion and Independent are set to true. The result of the execution, i.e. table ProcessInstanceLog is shown in Figure 36 in Appendix D.4.

By observing the log, it is concluded that an instance of sub process of type Reusable subprocess is logged as a separate process instance. This will impact the calculation of performance indicator related to that child process. The reference to the parent process instance is stored in column *parentprocessinstanceid*.

3 Swimlane

In jBPM 6.0.1, swimlane has no execution semantics, but rather only act as a visualization means to group tasks based on the actor. Thus further implementation does not consider swimlane.

Temporal perspective

For information about time, table *ProcessInstanceLog* stores the start time (column *start_date*), the end time (column *end_date*) and the duration (column *duration*) of process instance. Duration is basically the difference between the start time and the end time, stated in milliseconds. This information can be used to calculate throughput time of process. Observation shows that aborted process instance will also have end time, thus selection based on process instance status must be made possible.

Table *NodeInstanceLog* has *log_date* for time-related field which is the time when a node is entered (*type*=0) or a node is exited (*type*=1). This time-related information can be used to calculate the sojourn time of a task. However, we can make use of table *BAMTaskSummary* and *TaskEvent* to calculate task-related and time-related indicators since it provides more complete time information regarding human task.

Table *VariableInstanceLog* has column *log_date* to store the time when a variable value is changed.

3.2.4 Data Types

When defining data elements, the users can specify the data type for them, either standard data type or custom data type. KIE workbench provides the following standard data types for variable declaration: Integer, Boolean, String and Float. Custom data type is a user-defined data type which contains one or more attributes and can be considered as an object. This custom data type is declared in Data Modeler module. For example, a data type *Student* may contain attribute *id*, *name*, *dateOfBirth*, *nationality*. For custom data type, jBPM allows the following data type declaration for each attribute: BigDecimal, BigInteger, Boolean, Byte, Date, Double, Float, Integer, Long, Short and String.

To see how custom data type instance is stored in table *VariableInstanceLog*, the following scenario is tested.

Scenario 5 Observing the impact of using custom data types to table VariableInstanceLog

A process model is used (see Figure 30, in Appendix D.1) where in task1, a variable called student with type Student is modified by user. This data type contains attributes as follows:

```
id: Integer
name: String
```

The observation of execution result in table *VariableInstanceLog* concludes that jBPM (by default settings of KIE workbench) does not store the actual value of variable instance in the database. Instead, it refers to a record in another repository (not in the database) by specifying the identifier. Figure 12 shows the example of log entries for this case.

VariableInstanceLog

id bigi	oldvalue character varying(255	processinstanceid bigint	value character varying(255)	variableid character varying(255)
38		26	org.jbpm.masterthesis.Student@1901104	student
39	org.jbpm.masterthe:	26	org.jbpm.masterthesis.Student@138b395	student

Figure 12 Example entries in table *VariableInstanceLog* for custom data type

Further investigation shows that additional settings are required to store and extract this variable value (planet.jboss.org, 2014). In this master project, we limit the data elements type to be used in the care pathway workflow are the standard ones, i.e. Integer, Boolean, String, Datetime and Float.

3.3 Monitoring Variances

Variances are common in a high-risk sector like healthcare. Vanhaecht outlines that it is part of process indicator and the monitoring is suggested only in the key interventions of the pathway (Vanhaecht, K., & Sermeus, W., 2003). We distinguish this monitoring since it is a special case of pathway process monitoring and is important for care pathway evaluation and follow up. Several possible scenarios causing variances are:

1. Patient does not exit from normal happy path of care pathway due to referred to another protocol or die in the middle of treatment.
2. Early hospital discharge due to improvement on patient condition, and hence not all activities in pathway are performed.
3. Unusual treatment activities, in which normal pathway is not followed, due to emergency and unpredictable patient situation.

The analysis is performed to see what can be measured from implementing exception handling in pathway model. Exception handling is considered as a way to model predicted scenarios that can cause variance to occur in the pathway. In the implementation of model in BPMN, it involves the use of various types of events. To see how events can be implemented, we observe some model fragments that use events, i.e. as shown in several exception handling patterns (Lerner, et al., 2010). These patterns are described in Table 18 in APPENDIX E.

By observing the sample process model fragments, it is concluded that:

One way to monitor variance is by observing the occurrence of events that are implemented in pathway model. Interesting measure that can be derived from implementing event in pathway model is the number of occurrence of an event for a patient. This measurement is possible to be performed in jBPM since it stores all nodes (including events) that are actually executed during pathway runtime, i.e. in table *NodeInstanceLog*.

3.4 Identifying Care Pathway Performance Measures

As concluded from analysis of the Leuven compass, it is necessary to identify possible measures that compose an indicator with respect to the compass and the information available in the jBPM log. Note that we do not identify indicators but rather on measures that can be combined to create indicator formula (see the definition of composite indicator in Chapter 3.1 and the difference between measure and indicator in Chapter 2.7). The identified measures are described as follows.

1. **Number of patient**
   This measure can be associated with the number of instance, since an instance of pathway represents a patient. Number of patient is used in many indicators with different requirements. The requirements are specified as constraints to select patient records based

on condition on pathway, data element, intervention, and event. For each type of condition, it is described as follows.

a. Pathway condition

This condition specifies what constraint must be met in pathway level. This condition is defined based on the information available in table *ProcessInstanceLog*. The parameters for this condition are specified in Table 2.

Table 2 Parameters for pathway condition

Parameter	Operation	Type	Description
pathway_start _date	=, >, >=, <, <=	Datetime	Date of patient arrival
pathway_end_date	=, >, >=, <, <=	Datetime	Date of patient discharge
pathway_duration	=, >, >=, <, <=	Integer	Length of hospital stay

Each parameter can be declared multiple times to accommodate for example time interval. In jBPM log, duration is stored in millisecond.

b. Data element condition

This condition specifies what requirements must be met on data elements. jBPM only stores data elements (which are unique) that are defined globally for the process model (or global variable in jBPM terminology) in its execution log. The locally defined data elements, i.e. defined in the tasks, are not stored in the log if they are not mapped to global data elements. To define this condition, we assume that the users understand the declaration of global variables OR are facilitated with the documentation of the mapping between global and local variables and their corresponding tasks. This condition is defined based on the information available in table *VariableInstanceLog* For each declaration of this condition; the parameters are specified in Table 3.

Table 3 Parameters for data element condition

Parameter	Description
data_element_name	Name of the data element
operator	Operation that is used to compare data_element_name and data_element_value. The possible value for this parameter are =, >, >=, <, <=
data_element_value	A value to be compared
data_type	Data type of the data_element_name_. The possible values are STRING, NUMERIC and DATETIME

This condition can be declared multiple times. However, each declaration involves all parameters above.

c. Intervention condition

This condition specifies what condition must be met regarding interventions/tasks being performed in the pathway. This condition is defined based on the information available in table *BAMTaskSummary.* The parameters are specified in Table 4.

Table 4 Parameters for intervention condition

Parameter	Operation	Type	Description
schedule_date	=, >, >=, <, <=	Datetime	The date when the intervention is scheduled
start_date	=, >, >=, <, <=	Datetime	The date when the intervention actually takes place
end_date	=, >, >=, <, <=	Datetime	The date when the intervention ends
duration	=, >, >=, <, <=	Integer	The duration of the intervention, from start date until end date.

An intervention might be performed multiple times. However, in this thesis we ignore aggregate calculation e.g. average value or execution position, e.g. the first one executed or the last one executed. Any occurrence of an intervention will be evaluated.

d. Event condition

This condition specifies the requirement for any events that might occur in the pathway and are specified in process model. This condition is defined based on the information available in table *NodeInstanceLog.* The parameters for this condition are described in Table 5.

Table 5 Parameters for event condition

Parameters	Operation	Type	Description
number_of_occurrence	=, >, >=, <, <=	Integer	This filter specifies how many times an event occurs

It is possible to declare multiple conditions for single event_id to accommodate interval. The information about event_id can be found in the XML file of the process model.

2. Number of execution of an intervention

This measure is required as the multiplier in financial domain in pathway compass and in monitoring of variance. It is defined as how many times a particular intervention is performed for each patient. The parameter for this measure is *intervention name*. This parameter equals to property "Name" of a task in the process model. In addition to that parameter, the users can specify the conditions as described in measure #1, *number of patient*. Interesting values that can be derived from this measure are average, minimum, maximum and summation of values.

3. Number of event occurrence

This measure is required in monitoring variance since events are found mainly in pathway exception handling. It is defined as how many times particular event is triggered for each patient. The parameter for this measure is *event id*, i.e. the identifier of event. In jBPM, *event id* can be found in the XML file of the process. In addition to that parameter, the users can specify the conditions as described in measure #1. Interesting values that can be derived from this measure are average, minimum, maximum and summation of values.

4. **Data element value**
   This measure represents the value of a data element. Since data elements value can be changed in multiple tasks or in repeated task, there are several possible values that can be derived from single instance, i.e.: the latest value and the initial value, the minimum value, and the maximum value. In addition, if the data elements type is numeric, more values can be derived from single instance, i.e. summation and average value. The users can specify the conditions as described in measure #1, *number of patient*. Interesting values that can be derived from this measure are average (numeric), summation of values (numeric), minimum and maximum.

5. **Time difference between two interventions**
   This measure requires two identifiers of intervention with respective start times as input parameter. It is formulated as

   $$\Delta t(\text{intervention 1}, \text{intervention 2}) = |\text{starttime}_{\text{intervention 1}} - \text{starttime}_{\text{intervention 2}}|$$

   The users can also specify the conditions as described in measure #1, *number of patient*. Interesting values that can be derived from this measure are average, minimum and maximum.

6. **Throughput time of pathway**
   This measure can be associated with the length of hospital stay. It is defined as the time difference between the start event of pathway and the end event of pathway. The users can also specify the conditions as described in measure #1, *number of patient*. Interesting values that can be derived from this measure are average, minimum and maximum value.

7. **Waiting time of an intervention**
   This indicator is important to see for example, how long a patient waits for an intervention. It is defined as the time difference between scheduled time of an intervention and its actual start time. The users can specify the conditions as described in measure #1, *number of patient*. Interesting values that can be derived from this measure are average, minimum and maximum value.

8. **Execution time of an intervention**
   This indicator is important to see how long an intervention takes place. It is defined as the time difference between actual start time and end time of an intervention. The users can specify the conditions as described in measure #1, *number of patient*. Interesting values that can be derived from this measure are average, minimum and maximum value.

An intervention might be performed multiple times in a pathway instance, for example in arbitrary loop pattern. In this project, the design and implementation of measure *time difference between two intervention, waiting time* and *execution time of an intervention* consider only the first occurrence of the intervention.

To test whether classification of measures above can support various types of care pathway indicator, the list of KPI of Unstable Angina pathway identified in (Vermeulen, 2013) is used. Note that the formulation highly depends on the modeling style i.e. how artifacts in the process specification are implemented, whether as data elements, activities or events.

The first indicator is *patients admitted to the CCU or EHH* which is described as *percentage of patients that are admitted to the CCU or EHH*. It is a composite indicator formulated as number of patients (measure #1) who are admitted to the CCU or EHH divided by number of patients (measure #1) who come for admission process.

The formulation of other Unstable Angina KPIs is explained in detail in APPENDIX F. In conclusion, the classification of measure above can formulate all UA KPIs except for one process indicator (due to insufficient information provided) and team indicator. The identified performance measures will be the basis for the design of formalization in the next chapter.

Chapter 4

Design and Implementation

Based on the requirement analysis in Chapter 3, especially on the identified pathway measures, the formalization of performance indicator is designed and the measurement system is implemented. This chapter covers the proposed design of indicator formalization and implementation details of the performance measurement system.

4.1 Proposed Formalization Design

In many business settings, performance indicators are expressed in natural language. Consequently they are not machine executable since the grammar is ambiguous. To make the machine, i.e. the performance measurement system, understand what instructions must be performed, a formalized definition and processor of performance indicator is necessary. Hence the grammar is defined and no longer ambiguous.

Measuring performance indicator is basically passing queries to data source and retrieving the query result. jBPM stores its history logs in a relational database and can be browsed using SQL queries. Thus performance indicators principally can be formalized as SQL queries. However, as explained in chapter 2.6, SQL has a limited capability in processing rows. On the other hand, performance indicators might combine multiple measures, i.e. multiple query results with arithmetic operations (e.g. division in ratio-based indicators) to produce an indicator value. This leads to an idea to use mathematical expression to formalize performance indicators. The expression combines arithmetic operations and query formulation.

However, combining query formulation leads to another problem. The users still have to face query complexities, e.g. determining which tables to join, which column to select and what function should be applied and how to apply the function. Thus a complexity reduction is necessary i.e. by introducing a higher abstraction of SQL. The related table(s) for each measure in combination with four types of conditions is given in Table 6.

Table 6 Related table(s) for each measure and condition

Measure	Related table(s)
Number of patient	ProcessInstanceLog
Number of execution of an intervention	ProcessInstanceLog, BAMTaskSummary
Number of event occurence	ProcessInstanceLog, NodeInstanceLog
Data elements value	ProcessInstanceLog, VariableInstanceLog
Time difference between two interventions	ProcessInstanceLog, BAMTaskSummary
Throughput time of pathway	ProcessInstanceLog

Waiting time of an intervention	ProcessInstanceLog, BAMTaskSummary
Execution time of an intervention	ProcessInstanceLog, BAMTaskSummary
Condition	**Related table(s)**
Pathway condition	ProcessInstanceLog
Data element condition	VariableInstanceLog
Intervention condition	BAMTaskSummary
Event condition	NodeInstanceLog

There are several higher level abstraction techniques of SQL available, for instance Java Persistence Query Language (JPQL) as part of Java Persistence API, Java Object Oriented Querying (JOOQ), and Language Integrated Query (LINQ) in .NET framework. Those techniques are integrated with particular programming language and hence can benefit from the capability of the language, i.e. Java for JPQL and JOOQ and various .NET programming languages for .NET framework. Consequently, the syntax for specifying queries resembles the programming languages in combination with SQL.

Defining performance indicators in which relational database is used as data source in SQL or programming-language-like techniques above might be difficult to understand for non-programmer persons. On the other hand, care pathway performance indicators are communicated across multiple functions, not only among IT persons as the query formulator and executor but also among medical teams. Moreover, the abstraction method above does not solve complexity issue of the indicator queries. Hence a notation that is easily learnable by different discipline in healthcare sector with limited or no programming expertise, especially in care pathway domain, is required. In this thesis, an expression model is proposed to formalize care pathway performance indicator. The idea is to develop a limited set of SQL query template representing eight measures as identified in chapter 3.4 and to provide a means for the users to specify the parameters for the queries in a defined format.

The structure of jBPM logs is considered static, i.e. no special treatment for different business processes, and the scope of measurement is limited to the Leuven compass domains (implied by eight measures identified). Given these facts, it is advantageous to define a care-pathway-specific notation to define queries for performance measurement purpose. The advantages of this scoping are explained as follows.

Concise. Since the expression contains limited and small number of symbols, i.e. only the necessary ones, it is brief but covers all requirements for care pathway performance measurement.

Minimizing error. The queries for eight measures are predefined (not necessarily simple). The users only need to specify the parameter of predicate for the query and does not need to consider what relations or columns are involved. Thus the error can be minimized.

Learnable. The expression can be designed to use domain-related terminologies, in this case care pathway. As such, the users are expected to learn and interpret the grammar of the expression easily.

The development of the proposed expression model follows the development phase of domain-specific language (DSL) as shown in Figure 13. (Mernik, Heering, & Sloane, 2005)

Figure 13 Domain-specific language development phases

Note that some steps in the development phases above overlap with the research methodology of this master project. For example, the analysis of problem domain is already covered in Chapter 3. For analysis phase, the discussion will conclude the analysis of care pathway by providing the metamodel for performance indicator. The implementation phase can be considered as implementation of KPI formalization module which is only one part of the performance measurement system. Each stage is explained as follows.

4.1.1 Decision

In this phase, the motivation to develop a domain specific language is formulated. Based on the decision patterns outlined in (Mernik, Heering, & Sloane, 2005), the decision to formalize care pathway performance indicator is to enable the development of indicators by users with less domain and programming expertise or users with some domain but no programming expertise. In more specific, the decision is driven by unavailability of care pathway performance indicator notation, thus it is necessary to add user-friendly notation to an existing API or convert an API into DSL (Mernik, Heering, & Sloane, 2005). In the context of this project, a user-friendly notation to an existing API, i.e. SQL is proposed.

4.1.2 Analysis

In the analysis phase, the problem domain is identified and the domain knowledge is gathered. The inputs for this phase are various explicit and implicit domain knowledge, such as technical document and knowledge given by domain experts. The output of this step can be in various forms, but basically consists of domain-specific terminology and semantics in more or less abstract form. This phase is related to knowledge engineering topic, such as knowledge capturing, knowledge representation and ontology development (Mernik, Heering, & Sloane, 2005).

As mentioned earlier, this phase is already discussed in Chapter 3 which results in eight performance measures and their definitions. From those performance measures, the metamodel of care pathway performance indicator is created to represent domain model (see Figure 14).

The metamodel is explained as follows. A performance indicator combines numbers, arithmetic operators, and performance measures. For each performance measure, it has an aggregate function and a measure parameter. A measure parameter specifies which measure is observed (i.e. measure type) which pathway is involved (i.e. pathway identifier), for what status (i.e. pathway status), and additional parameters used by particular measure (i.e. specific parameter, if any). In addition, a measure parameter also defines what conditions must be met, specified in four types of conditions. For each condition type, it contains one or more condition item(s). Note that for the sake of readability, the metamodel above does not include various types of aggregate function and different types of condition item for each condition type.

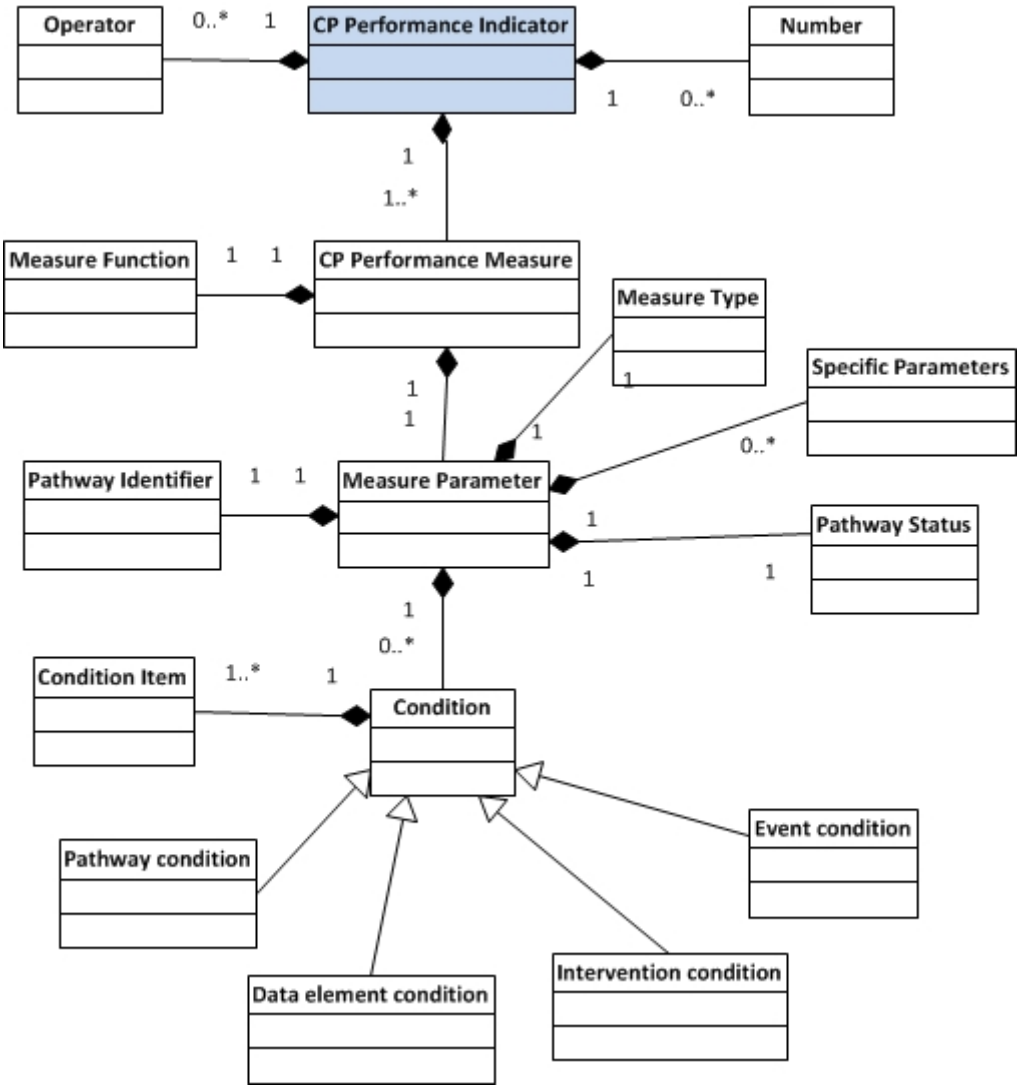


Figure 14 Care pathway performance indicator metamodel

4.1.3 Design

In the design phase, the abstract formalization model is brought into concrete syntax design. There are two dimensions being considered in the design phase, i.e. the relationship between the developed DSL and existing languages, and formal nature of the design description (Mernik, Heering, & Sloane, 2005).

Regarding first dimension, the language exploitation pattern is adopted, where existing language(s) is used as the source of adoption. The performance indicator expression is inspired by arithmetic expression which is usually written in the infix style, i.e. the operator is written between the operands, for example a + b. The formalization of measure is inspired by SQL abstraction that specifies parameters for the query template. In this project, Java's *.properties* file, a file that is used to store configuration parameters for Java application, is used as inspiration to specify parameter. For each line in the file, it defines a parameter name and a parameter value. The value is assigned to the parameter using equality sign (=). However, in this project, the parameter specification is not defined per line, but separated using a separator character.

Considering the formal nature of the design description, the semi-formal design specification is applied in the project. In informal design, the specification can be formulated in natural language and may include illustrative DSL programs. In formal design, the specification is written in formal method, e.g. regular expression and grammar for syntax definition, and attribute grammar, rewrite systems, and abstract state machine for semantic definition (Mernik, Heering, & Sloane, 2005). In this thesis, grammar for expression is used as syntax specification and the semantics will be explained in informal way, i.e. in natural language. Some semantic definition might be given in the explanation of the syntax. The design specification of the proposed expression is explained as follows.

DEFINITION

Expression is defined as "*Any mathematical calculation or formula combining numbers and/or variables using sums, differences, products, quotients (including fractions), exponents, roots, logarithms, trig functions, parentheses, brackets, functions, or other mathematical operations. Expressions may not contain the equal sign (=) or any type of inequality*." (Mathwords, 2014).

An example of expression is (a * b) / (c + d) where a, b, c and d are variables and involves multiplication, summation, and division operation. Using arithmetic expression ensures high level of freedom for users to define performance indicators.

As defined by indicator metamodel (Figure 14) the proposed expression limits the definition of expression above only to accommodate the following components:

1. Number, can be integer or real number
2. Function, is defined as named instruction that has particular arity (number of input parameters) and returns a particular type of output. However this project will only implement six functions: AVERAGE, SUM, MIN, MAX, COUNT, and COST. The first five functions are used to process returned records of measure query and are explained in Table 7. The format of the input string, which is also an expression, is defined differently from this definition and will be described later in this chapter. Function COST is used to get the cost of an intervention or a medical resource.
3. Arithmetic operations: addition (+), subtraction (-), multiplication (*), division (/).

Table 7 Functions applied to returned records

Function name	Description
COUNT	Return the number of record. For a record collection $R = \{r_1, r_2, r_3, \dots, r_n\}$, $COUNT(R) = n$
SUM	Return the total value of a column, if the column data type is numeric. For a record collection $R = \{r_1, r_2, r_3, \dots, r_n\}$, $SUM(R, an\_attribute) = \sum_{i=1}^{n} an\_attribute(r_i)$
AVERAGE	Return the average value of a column, if the column data type is numeric. For a record collection $R = \{r_1, r_2, r_3, \dots, r_n\}$, $for\ COUNT(R) > 0, AVERAGE(R, an\_attribute) = \frac{SUM(R, an\_attribute)}{COUNT(R)}$

	$for\ COUNT(R) = 0, AVERAGE(R, an\_attribute) = 0$
MIN	Return the maximum value of an attribute in the record collection, if the attribute data type is numeric or date time.
MAX	Return the minimum value of an attribute in the record collection, if the attribute data type is numeric or date time.

SYNTAX OF THE EXPRESSION

The format of the expression follows the format of infix-style arithmetic expression in which the operator is put between two operands. The syntax of the expression is depicted in Figure 15

```
<number_or_function1> <operator1> <number_or_function2> <operator2> …
```

Figure 15 Syntax for expression

Note that using braces "(" and ")" is allowed to define the order of calculation.

Creating an expression is performed by human as the actor. The input for this task is a performance indicator given in natural language. To create an expression from indicators given in natural language, the steps below are followed:

Step 1:
Break the indicator into parts which represent either a measure as mentioned in Chapter 3.4 or a number. A sample scenario of this step is in ratio-based indicator, where it is broken down into numerator and denominator.

Step 2:
Identify operations involved and its relation with the parts. In ratio-based indicator, division operation is involved. If it involves more complex operation, use braces to define the order of calculation when necessary.

Step 3:
For each part in step 1, if it is not a number, specify what function is used and what conditions must be satisfied (along with its parameter). Construct it using the syntax given in Figure 16. Note that open and closed braces are used to specify function parameters.

```
<FUNCTION_NAME>(<FUNCTION_PARAMETER1>,<FUNCTION_PARAMETER2>,...)
```

Figure 16 Format for declaring a function

Since this project only implements five functions as described in Table 7 and function COST and those functions take only one string parameter, the format of the function specification can be refined as shown in Figure 17. The input string parameter is put inside double quote characters.

```
<FUNCTION_NAME>("<FUNCTION_PARAMETER1>")
```

Figure 17 Function with single input string parameter

Function COST has one input parameter (String data type) and the format is specified as: `COST("<intervention_name_or_medical_resource_id>")`. The returned value is a real number representing the cost of intervention or medical resource per unit. For example, to retrieve the cost of an x-ray test, we declare `COST("x-ray test")`.

Step 4:
Finally, construct the expression by combining all parts, along with the operators and braces when necessary.

The input parameter for all functions in Table 7 is a string representing parameters to pre-defined query templates to pathway logs for each measure, in a defined format. For all measures, the format is given in Table 8.

The guideline to read the format throughout the document is given below:

a. The declarations inside "[" and "]" are optional and the rest are compulsory. Note that to define those optional elements, the "[" and "]" signs are not required.
b. "<abc>" means that it is a variable called "abc" and the value can be changed, without "<" and ">" signs.
c. "//" is the comment in the format and must not be declared in the real expression.
d. "…" means that the declaration can be continued.
e. Semicolon ";" is used to separate parameter declaration.
f. Curly braces "{" and "}" are used to group conditions.
g. "|" character is used to separate parameter in a condition.
h. "&&" is used to separate condition items within a condition type.

Table 8 Syntax for query parameters for pathway measures

No	Measure	Syntax	Functions
1.	Number of patient	measure='num_of_patient'; pathway_name='<pathway_name>'; pathway_status='<pathway_status>'; [<conditions>]	COUNT
2.	Number of execution of an intervention	measure='num_of_intervention'; pathway_name='<pathway_name>'; pathway_status='<pathway_status>'; intervention_name='<intervention_name>'; [<conditions>]	AVERAGE, MIN, MAX, SUM
3.	Number of event occurrence	measure='event_occurence'; pathway_name='<pathway_name>'; pathway_status='<pathway_status>'; event_id='<event_id>'; [<conditions>]	AVERAGE, MIN, MAX, SUM
4.	Data elements value	measure='data_element_value'; pathway_name='<pathway_name>'; pathway_status='<pathway_status>'; data_element_name='<data_element_name>'; data_type='<data_type>'; value_type='<value_type>'; [<conditions>]	MIN, MAX, SUM and AVERAGE (SUM and AVERAGE are only applicable for numeric data type)
5.	Time difference between two interventions	measure='time_between_interventions'; pathway_name='<pathway_name>'; pathway_status='<pathway_status>'; intervention1_name='<intervention1_name>'; intervention2_name='<intervention2_name>'; [<conditions>]	AVERAGE, MIN, MAX
6.	Throughput time of pathway	measure='pathway_throughput_time'; pathway_name='<pathway_name>'; pathway_status='<pathway_status>';	AVERAGE, MIN, MAX

No	Measure	Syntax	Functions
		[<conditions>]	
7.	Waiting time of an intervention	measure='intervention_waiting_time'; pathway_name='<pathway_name>'; pathway_status='<pathway_status>'; intervention_name='<intervention_name>'; [<conditions>]	AVERAGE, MIN, MAX
8.	Execution time of an intervention	measure='intervention_execution_time'; pathway_name='<pathway_name>'; pathway_status='<pathway_status>'; intervention_name='<intervention_name>'; [<conditions>]	AVERAGE, MIN, MAX

In workflow context, <pathway_name> can be associated with the name of business proces. <pathway_status> refers to the status of the process instance as described in Table 1. Intervention name refers to the task name. The information about task name can be found in property "Name" of a task in the process model.

For measure *Data element value*, <data_type> can be replaced with NUMERIC or DATETIME. It specifies the data type of the data element. Remember that data element value can be changed multiple times in the execution of process instance. Depend on the <data_type>, the <value_type> can be replaced with one of the following options.

FIRST. This option means that the returned data element value is the first value entered. It is valid for both NUMERIC and DATETIME.

LAST. This option means that the returned data element value is the latest value. It is valid for both NUMERIC and DATETIME.

MIN. This option means that the returned data element value is the minimum value among data element changes. It is valid for both NUMERIC and DATETIME.

MAX. This option means that the returned data element value is the maximum value among data element changes. It is valid for both NUMERIC and DATETIME.

AVERAGE. This option means that the returned data element value is the average value of all data element changes. It is valid for NUMERIC only.

SUM. This option means that the returned data element value is the summation of all data element changes. It is valid for NUMERIC only.

[<conditions>] can be replaced with four types of conditions mentioned in measure *Number of patient* in chapter 3.4. For each type of condition, the format and the explanation are given in Figure 18, Figure 19, Figure 20, and Figure 21 respectively. Date time data type is written in the YYYY-MM-DD hh:mm:ss format.

```
pathway_condition=
{
  //format
  param=’<parameter_name>’|op='<operation>'|value='<a_value>' &&
  //<parameter_name> can be replaced with a parameter in Table 2
  //the <operation> for each <parameter_name> can be seen in Table 2.
  //example: pathway start date between 1 Jan 2014 00.01 – 31 Des 2014 23.59
  param='pathway_start_date'|op='>='|value='2014-01-01 00:01:00' &&
  param='pathway_start_date'|op='<='|value='2014-12-31 23:59:00' &&
  ...
};
```

Figure 18 Format for declaring pathway conditions

```
data_element_condition=
{
  //format
  data_element_name='<a_data_element>'|op='<operation>'|value='<a_value>'|
  type='<data_type>' &&
  //possible value for <operation> and <data_type> can be seen in Table 3.
  //example: Integer-type data element blood_pressure value is between 130
  //and 180
  data_element_name='blood_pressure'|op='>='|value='130'|type='NUMERIC' &&
  data_element_name='blood_pressure'|op='<='|value='180'|type='NUMERIC' &&
  ...
};
```

Figure 19 Format for declaring data element conditions

```
intervention_condition
{
  //format
  intervention_name=’<an_intervention>'|param='<a_parameter>'|
  op='<operation>'|value='<a_value>' &&
  //in workflow context, pathway intervention is associated with task
  //possible value for <a_parameter> and <operation> can be seen in Table 4.
  //example: x-ray is performed in less than 15 minutes (or 900 seconds)
  intervention_name=’x-ray'|param='duration'|op='<'|value='900' &&
  ...
};
```

Figure 20 Format for declaring intervention conditions

```
event_condition=
{
  //format
  event_id='<an_event_id>'|param='number_of_occurrence'|op='<operation>'|
  value='<a_value>' &&
  //possible value for <operation> can be seen in Table 5.
  //example: error event with id 12345 occurs at least once
  event_id='12345'|param='number_of_occurrence'|op='>='|value='1' &&
  ...
};
```

Figure 21 Format for declaring event conditions

EXAMPLE
To Ilustrate how to create performance indicator expression, the following performance indicator for Unstable Angina pathway is used (Vermeulen, 2013).

Performance indicator: "Angiographic success (successful PCI <20% stenosis) (%)"

It is defined as percentage of PCI patients with < 20% rest stenosis in all lesions.

In addition to that definition, a pathway condition is added, i.e. for patients arriving between 1 January 2014 and 31 July 2014. It is assumed that PCI patient is a type of patient in Unstable Angina pathway and is implemented as a Boolean-type data element called *is_pci*. If it is set to true, the patient is a PCI patient. Otherwise, it is set to false. It is also assumed that the quantity of stenosis is implemented as a Numeric-type data element called *stenosis_percentage.*

Step 1: Break into parts
Part 1 is the numerator and described as PCI patients with < 20% rest stenosis in all lesions where PCI is attempted, for patients arriving between 1 January 2014 and 31 July 2014. Part 2 is the denominator and described as total number of PCI procedures in this hospital for patients arriving between 1 January 2014 and 31 July 2014. Both part 1 and part 2 represent measure *number of patient*.

Step 2: Identify operations
The operation involved is division, i.e. numerator/denominator

Step 3: Specify functions and conditions
For part 1 (numerator), the used function is COUNT since it counts the number of patients. The conditions involved are pathway condition and data element condition. For pathway condition, the parameter is *pathway_start_date* representing patient arrival date. For data element condition, the condition to be satisfied are *is_pci = true* and *stenosis_percentage<20*.

For part 2(denominator), the used function is also COUNT. The involved conditions is pathway condition and data element condition. For pathway condition, the parameter is pathway_start_date representing patient arrival date. For data element condition, the condition to be satisfied is *is_pci=true*.

Step 4: Construct expression
The constructed expression is given in Figure 22.

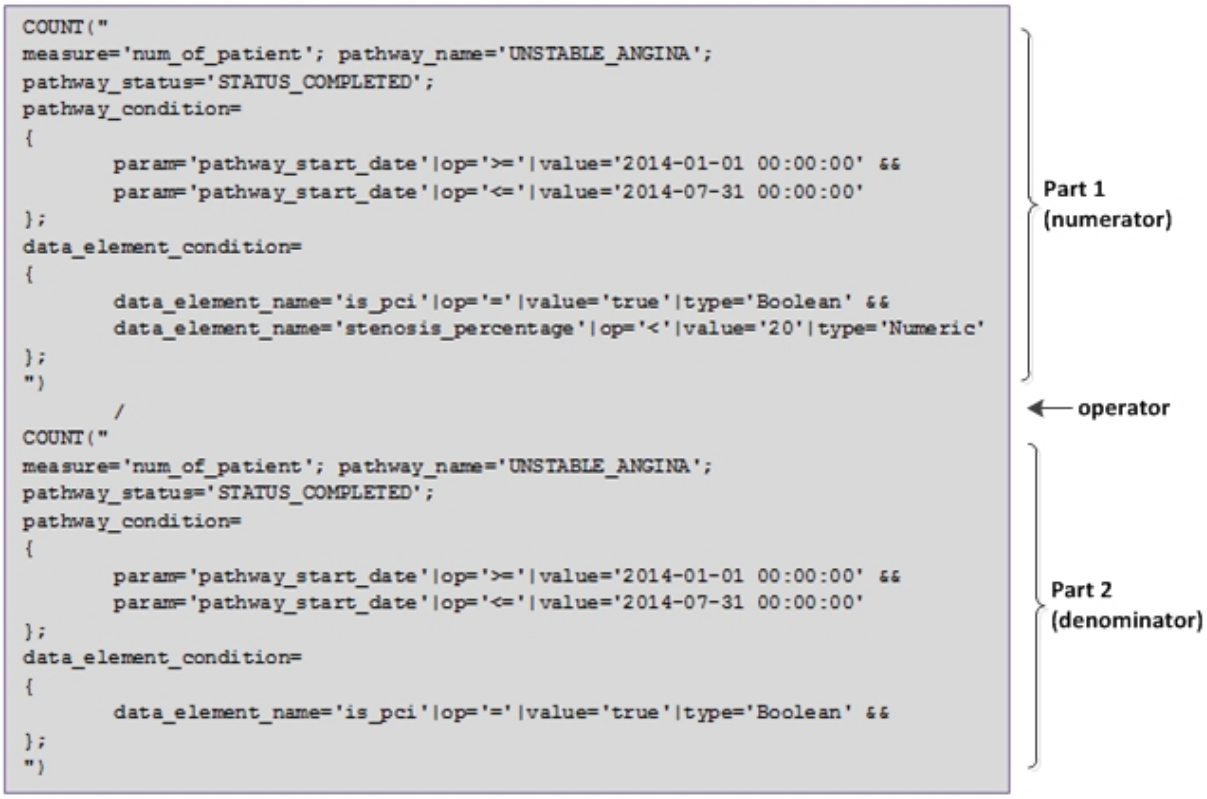


Figure 22 Example expression

SEMANCTICS OF THE EXPRESSION

The semantics of the expression is explained declaratively by the following rules.

1. Indicator expression is evaluated based on arithmetic expression evaluation which considers the order of calculation. For indicator involving multiple operands, all operands (including return value of function) must be numeric in order to be able to be evaluated. For indicator involving single operand (i.e. without any operator), the operand (e.g. function return value) can be numeric or date time.
2. Function COUNT, MIN, MAX can handle numeric and date time data types. Function AVERAGE and SUM can only handle numeric data type.
3. Defining particular measure type should declare measure-specific parameter(s) in addition to obligatory parameters (i.e. pathway_status and pathway_name) as defined in Table 8. Defining unnecessary parameter will be ignored.
4. All conditions are translated into AND conjunction which means that all defined conditions must be satisfied by returned collection of data. It is not always necessary to declare all four conditions, since an indicator might require only for example one or two conditions type.
5. Each measure along with its parameter and conditions are translated into SQL template as shown in Table 9. Note that the variables mentioned in the table refer to the variables used in the syntax declaration in Table 8 for measures and Figure 18, Figure 19, Figure 20, and Figure 21 for conditions.

Table 9 Query templates for measures and conditions

MEASURES

Number of patient:
`SELECT p.*`
`FROM processinstancelog p`
`WHERE p.processid = <pathway_name> AND p.status = <pathway_status>`
`//add conditions here`

Number of execution of an intervention
`SELECT p.*, getnumberofintervention(p.processinstanceid,<intervention_name>)`
`FROM processinstancelog`
`WHERE p.processid = <pathway_name> AND p.status = <pathway_status>`
`//add conditions here`
 	
function *getnumberofintervention* returns how many <intervention_name> is executed for a processinstanceid

Number of event occurrence
`SELECT p.*, getnumberofevent(p.processinstanceid,<event_id>)`
`FROM processinstancelog`
`WHERE p.processid = <pathway_name> AND p.status = <pathway_status>`
`//add conditions here`
 	
function *getnumberofevent* returns how many <event_id> occurs for a processinstanceid

Data elements value
if <data_type> = NUMERIC :
`SELECT p.*, getvariablevaluenumeric(p.processinstanceid, <data_element_name> <value_type>) val`
`FROM processinstancelog p`
`WHERE p.processid = <pathway_name> AND p.status=<pathway_status>`
`//add conditions here`
 	
function *getvariablevaluenumeric* return the numeric value of <data_element_name> for selected <value_type> for a processinstanceid
if <data_type> = DATETIME, the query template is similar to the template for NUMERIC but the function being used is *getvariablevaluedatetime* which returns the date time value.

Time difference between two interventions
`SELECT p.*, gettimedifference(p.processinstanceid, <intervention1name> , <intervention2name>) val`
`FROM processinstancelog p`
`WHERE p.processid = <pathway_name> AND p.status=<pathway_status> AND gettimedifference(p.processinstanceid,<intervention1name>,<intervention2name>) >= 0`
`//add conditions here`
 	
function *gettimedifference* returns time difference between two interventions. If one or both interventions in the parameter are not completed in the pathway, the function will return a negative value, thus omitted from the returned records.

Throughput time of pathway
`SELECT p.duration`
`FROM processinstancelog p`
`WHERE p.processid = <pathway_name> AND p.status = <pathway_status>`
`//add conditions here`

Waiting time of an intervention `SELECT p.*,getwaitingtime(p.processinstanceid, <intervention_name>)`

```
FROM processinstancelog p
WHERE p.processid = <pathway_name> AND p.status=<pathway_status> AND
getwaitingtime(p.processinstanceid, <intervention_name>) >= 0
//add conditions here
```

function *getwaitingtime* returns waiting time of <intervention_name> for a processinstanceid. If the intervention is not executed in the pathway, the function will return a negative number thus omitted from the returned records.

Execution time of an intervention

```
SELECT p.*,getexecutiontime(p.processinstanceid, <intervention_name>)
FROM processinstancelog p
WHERE p.processid = <pathway_name> AND p.status=<pathway_status> AND getexecutiontime
(p.processinstanceid, <intervention_name>) >= 0
//add conditions here
```

function *getwaitingtime* returns waiting time of <intervention_name> for a processinstanceid. If the intervention is not executed in the pathway, the function will return a negative number thus omitted from the returned records.

CONDITION

Pathway condition

```
AND <parameter_name> <operation> <a_value>
```

Event condition

```
AND getnumberofevent(p.processinstanceid, <an_event_id>) <operation> <a_value>
```

function *getnumberofevent* return the number occurrence of <an_event_id>
p.processinstanceid refers to processinstanceid of relation processinstancelog in the main query (i.e. measures)

Data element condition

```
AND EXISTS (SELECT * FROM variableinstancelog WHERE variableid=<a_data_element> AND
value <operation> <a_value>)
```

<data_type> is used to distinguish between numeric and date time datatypes and does not appear directly in the query but processed in the code.

Intervention condition

```
AND EXISTS (
SELECT * FROM bamtasksummary b
WHERE b.processinstanceid=p.processinstanceid AND b.taskname= <an_intervention> AND
b.<a_parameter> <operarion> <a_value>)
```

p.processinstanceid refers to processinstanceid of relation processinstancelog in the main query (i.e. measures)

4.1.4 Implementation

In this phase, the implementation strategy for designed DSL is chosen. In this project, *preprocessing* pattern is selected, in which the DSL constructs are translated to constructs in an existing language (the *base language*) (Mernik, Heering, & Sloane, 2005). The indicator expression is translated into SQL expression in order to be executable against care pathway log. There are two steps involved in the translation process: parsing arithmetic expression and parsing function parameter.

Parsing arithmetic expression will result three types of elements: number, operator and functions. In order for the expression to be able to be evaluated, the function must be first evaluated that will return a number. The parsing handlers and other concepts designed in the

design phase are implemented in Java classes and are depicted in a class diagram as shown in Figure 23.

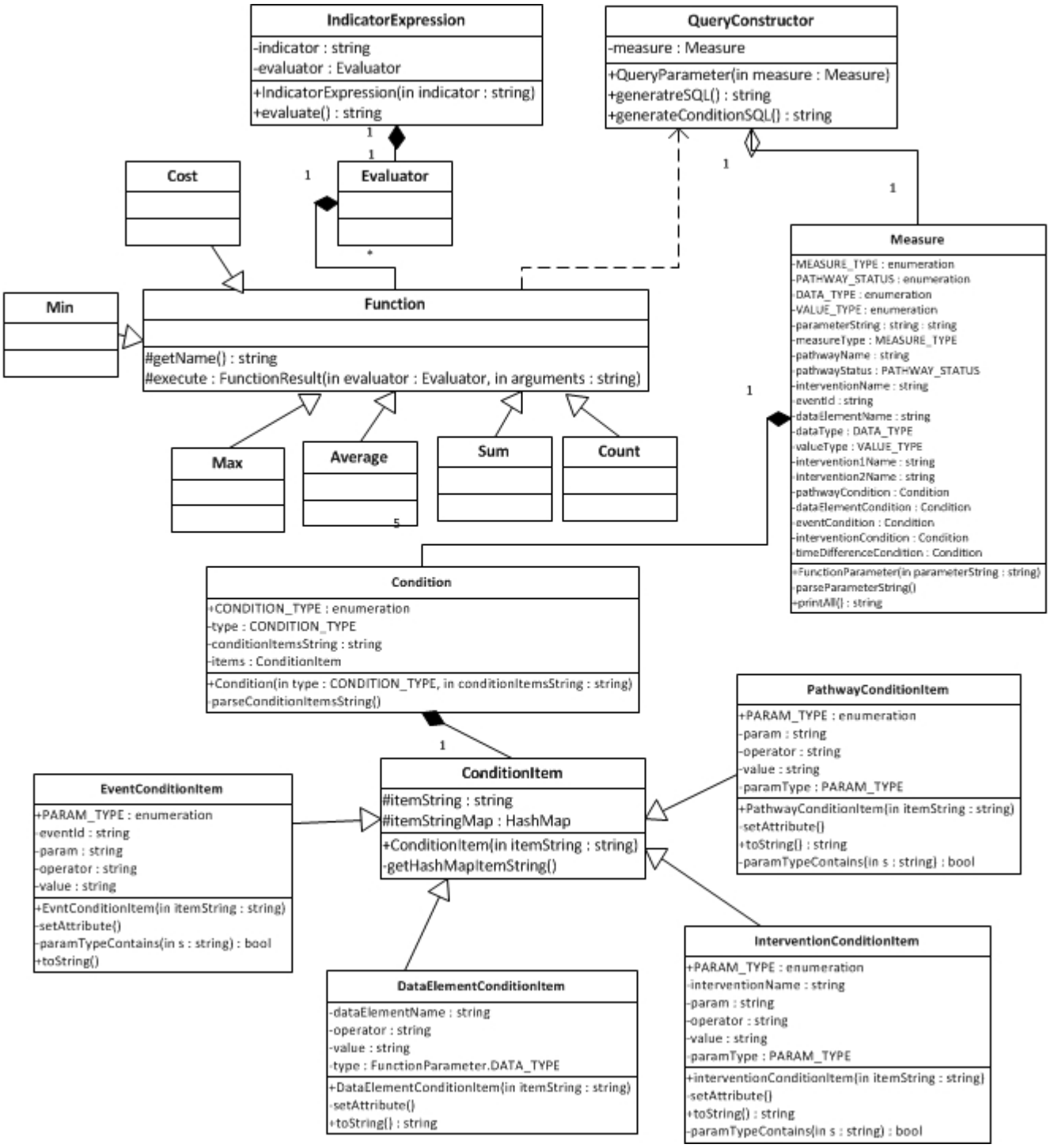


Figure 23 Implementation class diagram

4.1.5 Deployment

The deployment stage is not discussed in detail in (Mernik, Heering, & Sloane, 2005). This activity installs the software, in this case KPI formalization module, in the target machine to be able to run. In this project, the module is implemented as Java classes being part of web application of performance measurement system, i.e. not implemented as a library (.JAR file). Hence the deployment of the module is dependent to the deployment of the web application.

4.2 Implementation of Measurement System

Based on the formalization design, the performance measurement system is implemented. This chapter discuss about the architecture of the system, implementation environment and the functionalities of the system.

4.2.1 Architecture and Environment

The architecture of the system is shown in Figure 24. The performance measurement system is a web application hosted in a web server. It contains three main parts: the user interface where the user can interact with the system, KPI formalization module as the handler for performance measurement processing, and web module that controls the system scenarios. In supporting performance measurement, the web module calls required functions in KPI formalization module and passes produced query to BPMS execution logs stored in a relational database. In this thesis, a BPMS, i.e. jBPM 6.1.0.CR1 is used to implement care pathway. The formalization module also stores indicators created by the users in the indicator database in order to be re-used later.

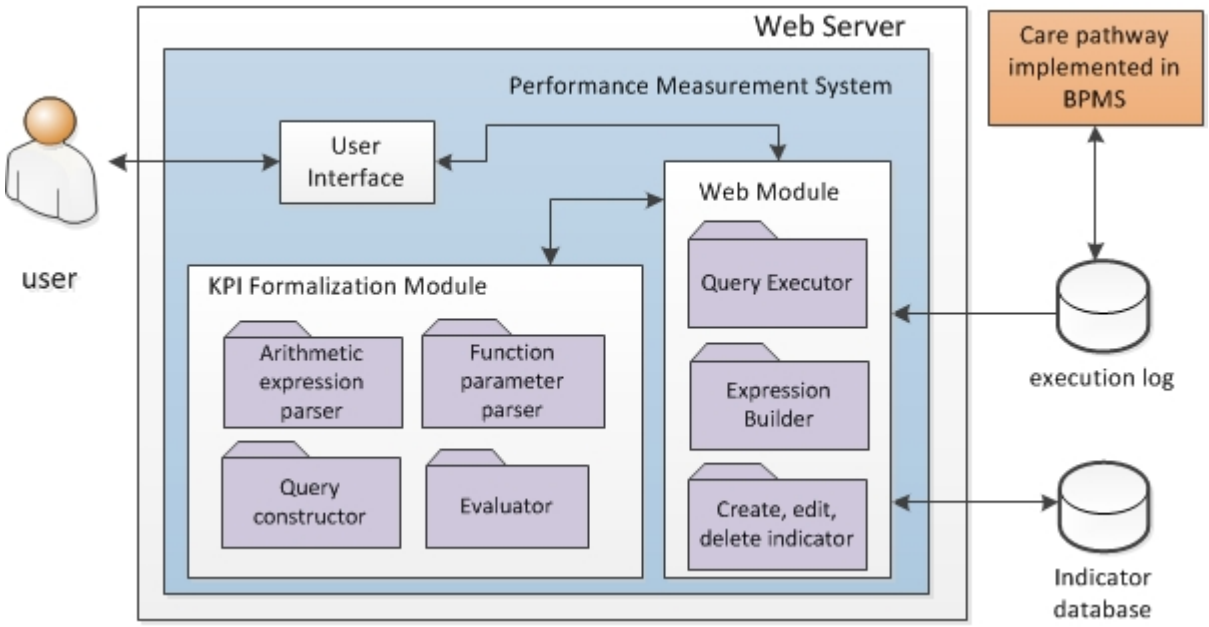


Figure 24 Architecture of the performance measurement system

The system is built under Windows 7 environment. The web application is built using Java Enterprise Edition using Netbeans IDE 8.0. Java Server Faces framework is used to develop the web pages. The web application runs on JBoss Application Server 7.1.1, similar to the one being used by jBPM 6.1.0.CR1. Jeval 0.9.4, an open source java-based library, is used to handle

arithmetic expression parsing and evaluation. The query constructor and function parameter parser is developed using Java. This project uses PostgreSQL 9.3 to manage jBPM database and indicator database.

The decision for the architecture is based on the following.

The measurement system is implemented as a web application to facilitate easy user access since it is accessible by intra/internet using a web browser without specific client tools and installation.

Web module is the heart of the system that acts as the controller of all system functionalities. It is separated from user interface to distinguish between logic and presentation. KPI formalization is a separate module focuses on modeling the formalization of KPI. By grouping KPI formalization module in a package, it opens possibility to distribute it as a library to be used in different application.

jBPM database is separated from indicator database, since update frequency of jBPM database is higher than indicator database due to operational use. By this separation, it is expected that the performance of jBPM database is not influenced by the measurement system. However, for the sake of simplicity to prove the proposed formalization, the implementation uses single database instance for jBPM and indicator database.

jBPM, especially its database, is considered as external system and the measurement system can only read it. In case the database is opted to be private, i.e. not directly readable by external applications except jBPM, a service oriented architecture can be adopted by developing the interface. Again, for the sake of simplicity to prove the proposed formalization, direct database access is chosen.

4.2.2 Functionalities

The implemented performance measurement system has four functionalities:

Query performance indicator
This feature enables the user to enter indicator expression and execute the constructed query to get the result. The interface is shown in Figure 42 in APPENDIX G.

Create, view, edit, delete, and update indicator value
This feature enables the user to store and manage indicator expression to be used later. In addition it also enables the user to re-calculate indicator value from the stored expression. The interface is shown in Figure 43 in APPENDIX G.

Build Measure
This feature enables the user to build expression of a measure by filling forms. It helps the user to minimize error in creating expression. The interface is shown in Figure 45 in APPENDIX G.

Create, view, edit and delete cost
This feature enables the user to store and manage cost of interventions and resources. The interface is shown in Figure 44 in APPENDIX G.

Chapter 5

Evaluation

In this chapter the evaluation of the proposed formalization and the performance measurement system is presented. The goal is to assess the quality of the developed system and to assess the degree of fulfillment of five criteria defined in the research objective. In this evaluation, some metrics are chosen from a model to evaluate the quality of software, i.e. the ISO/IEC 25010:2011 - Software Product Quality Model (see Figure 25) (ISO, 2011) (iso25000.com, 2014). The metrics written in red-bold are the ones considered in this evaluation. The selection of metrics takes into account that the developed measurement system is the proof of concept of performance indicator formalization. Therefore, the functionality is at the utmost attention. In addition, the selected metrics are related to the design choice of the architecture (see Figure 24) that needs to be evaluated.

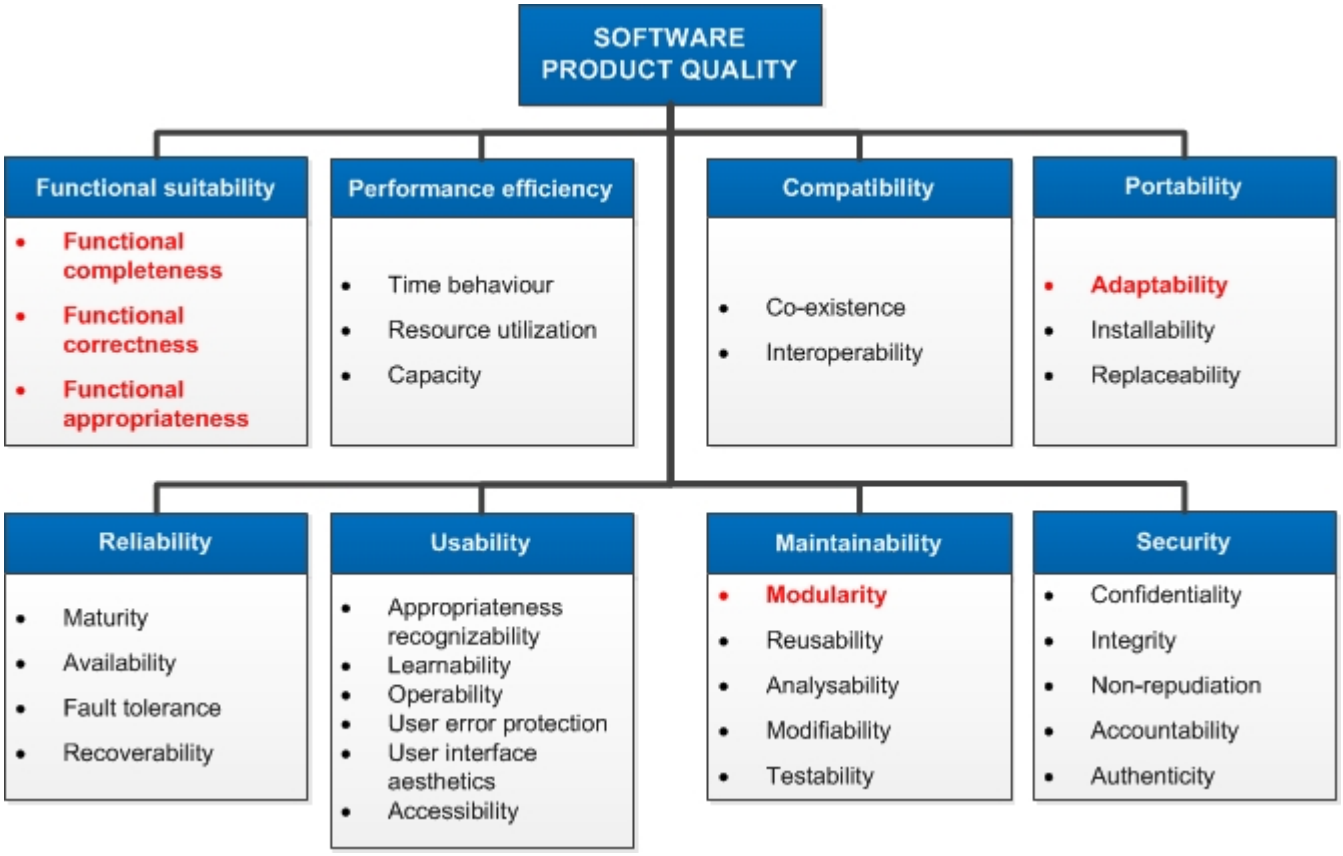


Figure 25 ISO/IEC 25010:2011 Software product quality metrics

Functional suitability determines the degree to which the product provides functions that meet the needs when used under specified conditions. Portability refers to the degree of effectiveness and efficiency with which a product can be transferred from one environment to another.

Maintainability measures the degree of effectiveness and efficiency with which a product can be modified by the intended maintainers. (ISO, 2011)

Chapter 5.1 discusses about the correctness of the system, chapter 5.2 discusses about functional appropriateness aspect, chapter 0 will discuss about the evaluation of other metrics, namely functional completeness, adaptability and modularity. Finally chapter 0 discusses about the degree of fulfillment of the criteria required in the research objective.

5.1 Functional Correctness

Functional correctness refers to the degree to which a product provides the correct results with the needed degree of precision (ISO, 2011). To evaluate this metrics, a case study in healthcare sector is used, i.e. weaning protocol. The overview of the evaluation is explained as follows.

The protocol is modeled in BPMN and is implemented in jBPM. Using the model, manual simulation is performed for 50 process instances with pre-defined test data (APPENDIX I). Several KPIs of the protocol are used to test the formalization by supplying it to the measurement system. The measurement system will return the results of the calculation of KPIs based on the execution log of the manual simulation. The correctness of the results is validated by doing direct queries to the execution log database and observation on test data.

5.1.1 Case Description

The definition of weaning protocol is obtained from master thesis by Boere which was conducted in Intensive Care Unit (ICU), Maastricht University Medical Centre+ (MUMC+). This process is said to apply to nearly all patients of the ICU department. There are two versions of weaning protocol, the < 72 hours ventilation and >72 hours ventilation, which are distinguished based on the nature of trial and error (Boere, 2013). In this thesis, the original, <72 hours version is used as a case study.

Even though weaning protocol might not a care pathway, it has many common characteristics with care pathway. There is a clear point in time where a patient starts and ends the protocol. The start time is when the patient arrives at ICU. The end time is when the patient is decoupled with the machine. Although the patients are not grouped based on similar disease, the grouping is clear, i.e. it is specialized to almost all patients who come to ICU. Finally, the protocol also documents, monitor, and evaluate variances and outcomes.

The flow of weaning protocol is described as follows (Boere, 2013):

"The mechanical ventilation process is initiated when a patient is not able to provide enough oxygen for the body. The process of oxygen supply is taken over by a mechanical ventilator. This machine provides the oxygen support and CO2 values during periods in which the patient is not self-sufficient. The Weaning process starts when the patient is connected to the ventilation machine and breathes with a tube. The name of this overall oxygen support is BIPAP (Bi-level Positive Airway Pressure). This means that the patient is fully supported by the mechanical ventilator and 100% dependable on the ventilator. When a patient arrives at the ICU, a target of the department is to discontinue the mechanical ventilating as soon as possible.

Of course the mechanical ventilator cannot be simply decoupled. Therefore the Weaning protocol has been developed to make sure that the decoupling is done under the right circumstances for the

patient. The main task of the nurses is to check for a specific set of pre-determined values (Note A), for example GCS (Glasgow Coma Scale) and a stable blood pressure. When a patient is brought to the ICU it is very unlikely that a patient is able to meet all threshold values. During the stay of the patient in the ICU with or without the use of specific medicines the right values can be met. When the values are met by the patient and a breathing trigger is given by the patient, the patient can go to the next state which is a different form of mechanical ventilating which depends more on the patient doing the work instead of the machine.

This breathing method is called ASB (Assisted Spontaneous Breathing). In this ventilation method the patient delivers a breathing pressure and the ventilator machine responds with oxygen and extra pressure to unload the ventilator attempt of the patient. Again, after some time specific values need to be checked by the nurses and met by a patient to have their breathing tube removed (Note C). Before the actual remove, the patient must remain stable (Note B) for a period of 30 minutes to make sure that the patient has the capability to breathe on his own again. When this uncoupling is performed, the process is finished and the patient breathes without support. During the process it can happen that patient needs to go back from the ASB ventilation to the BIPAP ventilation. This is the reason for the loops back that can be seen in the protocol."

The flow of weaning protocol can be seen in APPENDIX H. The corresponding BPMN model is given in Figure 26. The start event represents patient arrival in ICU.

Data Elements

The data elements defined in the model are obtained from Note A, Note B and Note C and are described in Table 10. Not all data elements are implemented in the process model due the limitation of the reference.

Table 10 Data elements in weaning process

Data Element Name	Data Type	Use
GCS_A	Double precision	GCS for Note A
PEEP_A	Double precision	PEEP for Note A
FIO2_A	Integer (0-100), %	FiO_2 for Note A
TACHYPNOEISCH_B	Integer	Tachypnoeisch forNote B
TACHYCARD_B	Integer	Tachycard for Note B
SPO2_B	Integer (0-100), %	SpO_2 for Note B
PO2_C	Double precision	PO_2 for Note C
PCO2_C	Double precision	PCO2 for Note C
PH_C	Double precision	pH for Note C
FIO2_C	Integer (0-100), %	FiO_2 for Note C
PEEP_C	Double precision	PEEP for Note C
GCS_C	Double precision	GCS for Note C
deviation_num	Integer	Number of deviations observed in task *30 min deviation check*

The data for Note A is entered in task *Check BIPAP*, for Note B in task *30 min deviation check finished* and Note C in task *Extubation criteria*. Basically, in order to continue the process, it is mandatory that all threshold values in all notes are satisfied in the corresponding decision point (gateway). However, since healthcare sector requires high level of flexibility, the decision to

continue the process is not implemented as a strict rule in the gateway by checking all threshold values. Rather, the decision to continue is made by human and is implemented using a Boolean data element representing the decision.

Roles. All tasks are performed by nurse, thus there is only one roles defined, i.e. nurse.

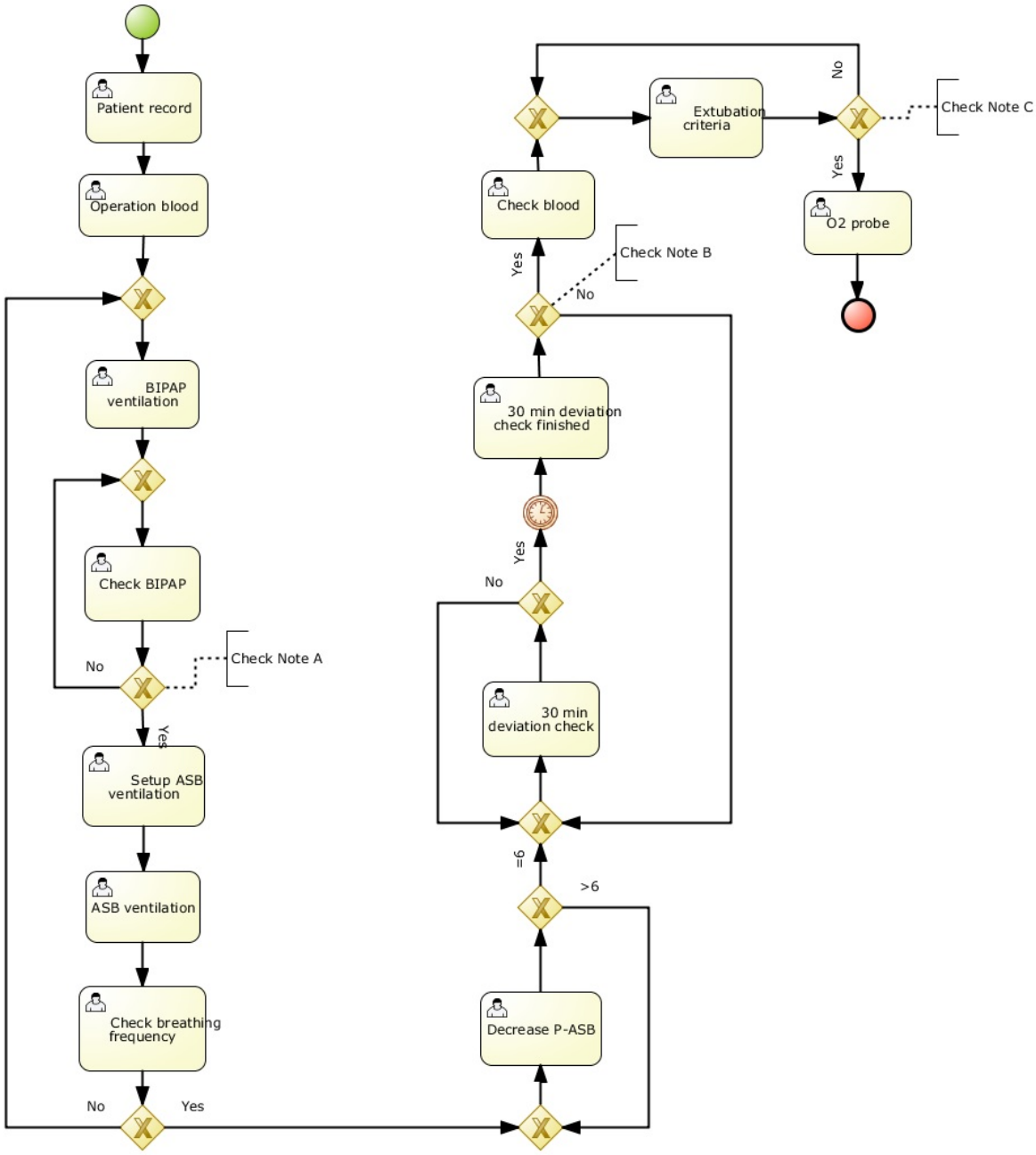


Figure 26 Weaning process in BPMN

5.1.2 Test Setup

Goal. The goal of the test is to prove that the proposed formalization can formalize and calculate KPI, and give the correct result.

Strategy and Data. The test is performed by running manual simulation of weaning process in KIE workbench for 50 instances. We consider that number is enough to perform validation. As a

comparison, Boere uses records from 617 patients arriving between 2009 and June 2012(approximately 3.5 years) (Boere, 2013). Automatic simulation in KIE workbench is not performed since the execution data is not stored in the database. The values of data elements for the test are generated randomly using spreadsheet. Even so, the value range still refers to the values found in the literature (Boere, 2013). The generated data elements values are given in Table 20 in APPENDIX I. Note that calculation of KPI related to temporal aspect does not represent real world, since all tasks are executed immediately. After the execution of weaning process, the validation is performed using some KPI of the process.

Test cases. From 50 cases, 5 cases will intentionally repeat *Decrease P-ASB* 3 times, i.e. in the decision point after task *Check breathing frequency*. 5 cases will repeat *Decrease P-ASB* for 4 times and 5 cases will repeat it for 5 times. The rest 35 will follow happy path of the model. There is no particular reason in choosing number of cases that repeat the task and number of repetition. They are arbitrary numbers that will be used to validate a KPI involving number of execution of an intervention.

User. A user called "nurse" with role "Nurse" is added to jBPM user list. This user will execute all tasks in the process model.

5.1.3 Result Validation

Validation is performed by specifying a list of weaning process KPIs, create KPI expressions using proposed formalization, and execute them in the measurement system. The results then compared with the data test (Table 20 in APPENDIX I) and direct query to database.

Key Performance Indicators

The list of KPIs in Table 11 is summarized from master thesis by Boere (Boere, 2013). In addition, the list has been confirmed to a researcher working on weaning process. Unfortunately, the list was not confirmed to process owner in hospital due to time constraint.

Table 11 List of weaning process KPI

Domain	Performance Indicator	Description
Process	Throughput time	The time a patient spends in ICU, from patient arrival until extubation.
Process	BIPAP time	The duration of BIPAP phase for a patient
Process	ASB time	The duration of ASB phase for a patient
Clinical	Time to reach 36° C	The time a patient needs to reach temperature 36' since his/her arrival.
Clinical	BIPAP threshold values fulfillment (NOTE A) (%)	Percentage of patients who meet BIPAP threshold values before continuing to ASB. Numerator: Number of patients who meet BIPAP threshold values. It refers to values in NOTE A in Figure 46 in APPENDIX H, i.e. GCS > 8, PEEP ≤ 8, FiO_2 < 50% Denominator: Number of patients
Clinical	Number of steps to decrease ASB	The number of steps performed to decrease P-ASB for a patient
Clinical	Extubation threshold values fulfillment (NOTE C) (%)	Percentage of patients who meet extubation threshold values before extubation.

		Numerator: Number of patients who meet extubation threshold values. It refers to values in NOTE C in Figure 46 in APPENDIX H,i.e. $PO_2 \geq 9$, $PCO_2 \leq 6.0$, pH 7.3-7.5, $FiO_2 \leq 40\%$, PEEP ≤ 8 and GCS > 8. Denominator: Number of patients
Clinical	Number of observed deviations within 30 minutes	The number of deviations identified in 30 minutes deviation check for a patient

From the list of indicators above, the corresponding indicator expressions are created as shown in APPENDIX J. One indicator expression cannot be built, i.e. *time to reach 36°C*. If the time to reach 36°C is implemented as a data element that store the duration, we can build the indicator. However this data element is not defined in the list of data element. Moreover, it is not reasonable to store the duration since the measurement of temperature is performed multiple times. A reasonable scenario is by storing the temperature data and the timestamp of measurement. This is the behavior of jBPM where all data elements value changes are stored. However, from the list of available measures, we still cannot build the indicator, since there is no measure to determine time difference between two change timestamps of a data element.

Result

During the test, some test cases do not follow planned cases. Only 4 cases that repeat task "Decrease P-ASB" 5 times, 6 cases repeat the task 4 times, and 5 cases repeat the task 3 times. However we consider this is still acceptable and does not influence the analysis of result. Using pre-defined test data, the result of the execution of the KPIs in performance measurement system is given in Table 12.

Table 12 Test result

Indicator	Result	Note
Throughput time		This value does not represent real life scenario since pathway is executed immediately. Manual query in PostgreSQL shows exactly similar result
Average	28467.1 seconds	
Min	24919.0 seconds	
Max	32080.0 seconds	
BIPAP time		Manual query in PostgreSQL shows exactly similar result
Average	65.2 seconds	
Min	11.25 seconds	
Max	242.68 seconds	
ASB time		Manual query in PostgreSQL shows exactly similar result
Average	31.87 seconds	
Min	4.49 seconds	
Max	94.05 seconds	
BIPAP threshold values fulfillment (NOTE A) (%)	10%	Manual check on the test data shows valid result, i.e. there are 5 records satisfy the constraint.
Number of steps to decrease ASB		Since there are 35 cases execute task *Decrease P-ASB* 1x, 4 cases 5x, 6 cases 3 times, the average is 1.88 times per patient.
Average	1.88 times/patient	
Min	1 time/patient	
Max	5 times/patiet	

Indicator	Result	Note
Extubation threshold values fulfillment (NOTE C) (%)	14%	Manual check on the test data shows valid result, i.e. there are 7 records satisfy the constraint.
Number of observed deviations within 30 minutes		Manual check on the test data shows valid result
Average	2.8	
Min	0	
Max	5	

In general, the tested KPIs can be formulated using proposed formalization, except one indicator involving time difference between two data element value changes. In conclusion, the measurement system is able to calculate the KPIs and gives the expected results, i.e. 100% correct.

5.2 Functional Appropriateness

This metric refers to the degree to which the functions of the product facilitate the accomplishment of specified tasks and objectives (ISO, 2011). We are interested in evaluating the degree of complexities that are perceived by the users by creating the queries for indicators manually and using proposed formalization. We argue that the less complex the task to form the queries, the higher the degree of accomplishment of measuring performance.

To evaluate this quality metric, four measures related to the complexity of SQL query structure are used (Brink, Leek, & Visser, 2007) and are explained as follows.

1. **Number of tables used.** This measure specifies how many tables are involved in a query. The complexity level rises along with the number of tables involved. In this thesis, we calculate the number of tables used not only from the FROM statement, but also from the nested queriues and function used (if any). A table might be used multiple times in a query, i.e. there are multiple relations specified that refers to single table. In this thesis, the number of tables used refers to the number of relations used.
2. **Number of nested queries.** A nested query is an SQL SELECT statement that is put in the predicate of any other SQL statement. Using nested query implies that the query complexity increases significantly for each deeper level. In this thesis, we ignore the level of the nested query to calculate this measure, but rater calculating the number of SELECT statement inside the main SELECT statement. We also treat function(s) used in a query as a nested query.
3. **Number of union queries.** This measure counts the number of UNION operation used in a query. The more it is used, the higher the complexity of a query.
4. **The number of joins used.** This measure counts the number of join operation in a query. The number of relations used in the FROM statement of a query implies the number of joins used. However, a relation might not appear in the FROM statement, e.g. when a function that uses tables is called or nested queries that use other relations. Therefore, we estimate the value of this measure as the value of measure *number of tables used.*

To execute the evaluation, each query of the performance measures in Table 9 will be observed to determine the value of four measures above. The calculated values also considers if the performance measures use condition in their specification. The result of this observation is given in Table 13.

Table 13 Query complexity of performance measures

Perforance Measure	Number of tables used	Number of nested queries	Number of union queries	The number of joins used
Number of patient	*1+p+d+e+i*	*d+e+i*	*0*	*1+p+d+e+i*
Number of execution of an intervention	*2+ p+d+e+i*	*1+d+e+i*	*0*	*2+ p+d+e+i*
Number of event occurrence	*2+ p+d+e+i*	*1+d+e+i*	*0*	*2+ p+d+e+i*
Data elements value	*2+ p+d+e+i*	*1+d+e+i*	*0*	*2+ p+d+e+i*
Time difference between two interventions	*2+ p+d+e+i*	*1+d+e+i*	*0*	*2+ p+d+e+i*
Throughput time of pathway	*1+ p+d+e+i*	*d+e+i*	*0*	*1+ p+d+e+i*
Waiting time of an intervention	*2+ p+d+e+i*	*1+d+e+i*	*0*	*2+ p+d+e+i*
Execution time of an intervention	*2+ p+d+e+i*	*1+d+e+i*	*0*	*2+ p+d+e+i*

Note for Table 13**:**

p : number of condition item specified in *Pathway condition*
d : number of condition item specified in *Data element condition*
e : number of condition item specified in *Event condition*
i : number of condition item specified in *Intervention condition*

From the table above we can conclude that the complexity of the queries of all performance measures increases linearly along with the number of conditions specified in the measure (except for number of union queries). Note that the values above are the values for single measure. If a performance indicator involves multiple measures, we can easily estimate the complexity of the indicator by adding up the complexity values of each measure involved.

The developed formalization does not eliminate the complexities of the queries in the context of the system. However, from the user perspective, the complexities are not perceived since they are handled by the system in the background, i.e. zero complexity with respect to four query complexity measures above. In conclusion, the formalization increases the degree of accomplishment of measuring performance, thus increase the level of functional appropriateness.

5.3 Other Metrics

In this section, the evaluation of the other selected metrics is discussed.

Functional completeness

This metric determines the degree to which the set of functions of the product covers all the specified tasks and user objectives. The analysis of the Leuven compass concludes that the formalization supports the measurement of all domains with the exception for team indicators and limitation for service indicator. Based on the definition of team indicator, measuring indicators belong to this type is not relevant in the context of care pathway process model. The pathway process model focuses on the patients' journey and does not focus on the relationship among team members. For service indicator, the formalization has a limited support, i.e. it only supports a simple single-question survey to rate the care services that can be implemented as a data element.

Adaptability

This metric determines the degree to which a system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments. Here we discuss about the degree of adaptability with respect to the selected development environments, i.e. programming language, database, and jBPM. By using Java as the development language of the system, it is expected that the level of adaptability is increased; since it can be installed in any environments with Java Virtual Machine is installed. jBPM allows the users to store the execution log in any major relational DBMS. However, since each DBMS has their specification for data type, the developed system is dependent to the selected database, i.e. PostgreSQL. A change in the database preference will need to modify the code and small part of the queries, even though jBPM uses similar schema for its execution log for all DBMS. Regarding jBPM, the BPMS is continuously developed and improved. If the newer versions uses similar schema for its execution log, we can say that the system is quite adaptable.

Modularity

This metric determines the degree to which a system is composed of discrete components such that a change to one component has minimal impact on other components. From the class diagram of the formalization module (Figure 23), we observe two classes that are extendable, i.e. class *Function* and class *ConditionItem*. With this structure, we can extend the formalization by adding new functions without influencing the existing ones, thus increasing the level of modularity. The class *ConditionItem* is not necessarily extended with new classes since the number of condition is fixed. Adding new measures will need to modify class *Measure*, i.e. by adding some code segments in the class definition, since measures are defined statically in the code (i.e. not a sub-class per measure). However, we consider that it does not have significant impact on the modularity of the system.

5.4 Evaluation of Formalization Criteria

The answer to the expected criteria of formalization in the research objective is explained as follows.

Comprehensive

The analysis shows that the formalization can facilitate all domains in the Leuven compass except one dimension, i.e. Team dimension. This is due to the definition of team dimension given by the guideline that measures the effectiveness of multidisciplinary team in doing their job. Since the care pathway workflow represents patient journey, the data for this indicator is not relevant to be collected in every pathway instance. Testing the formulation on KPI for Unstable Angina also shows potential use with respect to the Leuven compass.

Expressive

The adoption of arithmetic expression ensures high level of freedom for users to define indicator formula. Regarding expressiveness of the formalization in depicting the problem domain, eight measures identified are able to formulate most of tested indicators.

Generic

This criterion needs further test, since the evaluation involves only one process. However, by seeing the ability of the proposed formalization in describing indicator for weaning process and unstable angina, it is expected that the formalization can also support other pathways.

Intuitive

This criterion is related to the usability aspect of formalization and needs further test on the persons who will actually use the formalization. Due to time limitation, the test is not performed.

Executable

By developing a processor for the formalization, i.e. parser, query constructor and evaluator, it is executable against jBPM execution log and can be shown in the performance measurement system.

Chapter 6

Conclusion

This chapter discusses the conclusion drawn from the execution of research methodology. First sub chapter will provide the answer to the research objective and the research question, and discuss the contribution of this thesis. Second, it will discuss about the limitation of the proposed solution. Third, it will provide the direction for further research. This chapter closes by brief discussion about software technologies related to the solution.

6.1 Research Conclusion and Contribution

The answer to the formulated research question is explained as follows.

Research question: *Given a care pathway executable in a BPMS, i.e. jBPM, how can its performance indicators be formalized and calculated if The Leuven Clinical Pathway Compass is used as the guideline?*

The formalization is achieved by introducing a new notation representing care pathway performance indicator. It is inspired by the expressiveness of arithmetic expression in defining formula. The indicator expression combines eight types of performance measures, i.e. number of patient, number of execution of an intervention, number of event occurrence, data elements value, time difference between two interventions, throughput time of pathway, waiting time of an intervention, and execution time of an intervention. In addition to those measures, the constraints to the measures are also identified, i.e. four types of conditions: pathway condition, data element condition, intervention condition, and event condition. Both performance measures and conditions are identified based on the analysis of jBPM and The Leuven Clinical Pathway Compass. jBPM is the BPMS being used to test the implementation of care pathway.

To be able to calculate performance indicator, the formalization is brought into executable program module that can perform query to jBPM execution log. It contains parsers, query formulator and arithmetic evaluator. This formalization module is then used in the developed performance measurement system.

To evaluate the quality of the proposed formalization, a process model representing care pathway is used, i.e. weaning protocol. The evaluation result of functional correctness shows that the formalization can support performance measurement of weaning protocol and gives the expected results, even though one indicator cannot be modeled. This is due to the subjectivity in process modeling in term of style, strategy, and goal. For example, an artifact in the textual process description may be interpreted in many ways and is modeled in different ways e.g. as a data element or a condition in the gateways.

The evaluation of functional appropriateness concludes that the formalization can reduce the complexity of task perceived by the users in measuring performance. It is also concluded the clinical, process and financial domain in the Leuven compass are supported by the formalization while service domain is limitedly supported and team domain is not supported. From five criteria defined in the research objective, a criterion, i.e. *Intuitive*, needs further evaluation since it is related to the learnability aspect of the notation and requires user's interaction.

In general, the research objective is achieved. The contribution of this thesis is summarized as follows.

This project provides a new approach to formalize performance indicator and provides a performance measurement system for care pathway that is implemented in a BPMS, i.e. jBPM. Previously, it is difficult and time consuming to measure pathway performance due to difficulty in formulating performance indicator, complexity issues in the data source, and the availability of measurement system. Using the proposed formalization, the complexity is reduced, i.e. the users with limited programming or database knowledge are expected to be able to measure care pathway performance. This project demonstrates that continuous follow up on care pathway implementation is feasible in practice using the proposed framework.

6.2 Limitations

This research project has several limitations.

1. The identified measures cannot formulate relationship between two or more data elements of a pathway instance. This limitation is observed when the project is in the evaluation phase. For example, in weaning process P/F ratio of a patient can calculated by PO_2/FiO_2 (Boere, 2013), i.e. it is derived from two data elements. A new measure that enables calculation between data elements in one instance is open for development.
2. The identified measures cannot formulate time difference between two value changes of a data element. This is found in indicator *time until patient reach 36° C* in weaning process. Seeing jBPM log structure, especially in table *variableinstancelog*, a new measure for handling this issue is also open for development.
3. The requirement analysis is based only on literature review and does not involve future users. The test also uses data generated by manual simulation in jBPM. A further collaboration with medical parties is required to test the usability aspect of the proposed formalization.
4. The formalization translates conditions into AND conjunction which means that all query constraints must be satisfied. It is possible that some indicators use OR operation.
5. This project assumes that the users understand the declaration of global variables in the pathway model. A documentation explaining the mapping between local and global variables and their corresponding tasks can be created to assist the users in selecting data elements; to declare data element condition or data element value measure.

6.3 Future Research

Based on the research conclusion and limitation, the directions for future research are presented.

1. The limitations imply that the formalization is open for further extension, i.e. identification of additional performance measures and their implementation. Future research can also test the formalization in different BPMS environment, to see whether the classification of measures and conditions are applicable in different BPM systems. A better (computer) language processor library such as ANTLR can be used to process the grammar.
2. The formalization is possible to be implemented as a module in Dashbuilder, since it provides good visual reporting capability.
3. The formalization of sample indicators being used in the test (KPI for unstable angina and weaning process) assumes that some information in the description are implemented e.g. as data elements (due to process modeling subjectivity). Future research can investigate the process modeling guideline for care pathway with the goal to measure KPI.
4. The future research can investigate the possibility to integrate the proposed formalization of KPI with executable process model, from creating indicator expression until the presentation of the result.

6.4 Related Software Technology

Several software technologies are discussed to see the relationship with the proposed formalization and to identify the opportunity for improvement.

Dashbuilder

This web based tool provides performance monitoring and reporting capability to jBPM starting from version 6.0. The users are allowed to define their own KPI in the form of SQL query to jBPM database or any databases they would like to use and the system will show the result visually in the form of charts, tables, etc. In addition, it also accepts comma separated values (CSV) files as the datasource. The tool package that comes with jBPM is equipped with basic queries to table *ProcessInstanceLog* and *BAMTaskSummary*.

However, this tool only provides means, but does not solve complexity issues of the queries. The dashboard developers still have to learn the structure of the jBPM logs and how to formulate the queries. The proposed formalization has a potential to be implemented as a module in Dashbuilder to enable care pathway performance measurement. By combining them, it is expected that the users will have minimum efforts in defining KPI and benefit from the visualization power of Dashbuilder.

Drools Rule Language (DRL)

DRL is a language to define rule in Drools, a Business Rule Management System under the same project umbrella with jBPM. We are interested in discussing this language since there are some constructs in DRL that are able to filter a collection of objects and perform operations on it. This has similar scenario with the proposed formalization in which the collection of objects is the collection of pathway instance and the operations are implemented in some aggregation

functions. For more details about Drools, please refer to Drools Documentation (The JBoss Drools team, 2013).

A sample rule representing this issue is given below[1].

```
rule "Number of process instances above threshold"
when
  Number(nbProcesses : intValue > 1000)
  from accumulate(
      e:ProcessStartedEvent(processInstance.processId ==
        "com.sample.order.OrderProcess")
      over window:size(1h),
      count(e))
then
  System.err.println("WARNING: Number of order processes in the last
hour above 1000: " + nbProcesses );
end
```

The rule above can be interpreted as: When the number of started process for processId = "com.sample.order.OrderProcess" within last one hour is greater than 1000, print a warning message. In this case, the collection of object is of type ProcessStartedEvent (the started processes) and the filter is the processId. "accumulate" enables the rule to iterate over a collection of objects, executing custom actions, and return a result object at the end. Function "count" is one of several built-in functions for accumulate that enumerates the number of items in the collection. This declaration resembles (incomplete) declaration of indicator:
`COUNT("measures='num_of_patient';pathway_name='pathway name';")`
in the proposed notation.

Regardless the difference in the purpose of language between DRL and the proposed notation, some discussions can still be made about those two.

Drools works under object orientation paradigm, i.e. the *facts* that are stored in Drools' working memory are objects with attribute(s). It is possible that the collection of objects is of many classes (or *type* in DRL terminology) and a class has attribute(s) of other class. In this project, the items processed are records in multiple related tables in a database. At a glance, we can associate class with tables, class' attributes with table's columns. However, further investigation is required to see how class relationship is implemented in Drools. This investigation is important to assess whether it is possible (and if possible, easy) to formulate constraints in the query to the collection of object involving multiple classes (i.e. join operation in SQL) in DRL as required by formulation of KPI. In comparison to the proposed notation, the constraints are explicitly specified in four types of conditions.

For persons with limited programming knowledge, DRL might be difficult to understand. However, it has an "expanders" that allows the language to morph to user's problem domain, either in natural or domain specific language (The JBoss Drools team, 2013). The proposed notation is already in the form that represents care pathway domain.

[1] Taken from http://docs.jboss.org/jbpm/v5.1/userguide/ch16.html

Bibliography

Aboriginal Health & Medical Research Council. (2013). A literature review about indicators and their uses. *Aboriginal Health & Medical Research Council*. Sydney, Australia.

Aledo, V., Ballester, M., Franco, E., Alcaraz, A., Baldo, M., Prats, M., . . . Albasini, J. (2011). Evaluation of Clinical Pathway to Improve Colorectal Cancer Outcomes. *American Journal of Medical Quality*, 396-404.

Aledo, V., Pastor, B., Arenas, M., Alcaraz, A., Soto, A., Perello, J., . . . Albasini, J. (2008). Evaluation and Monitoring of The Clinical Pathway for Thyroidectomy. *The American Surgeon*, 29-36.

Boere, J.-J. (2013). *An analysis and redesign of the ICU weaning process using data analysis and process mining (Master's Thesis).* Eindhoven: Eindhoven University of Technology.

Brink, H. v., Leek, R. v., & Visser, J. (2007). Quality Assessment for Embedded SQL. *Seventh IEEE International Working Conference on Source Code Analysis and Manipulation* (pp. 163 - 170). Paris: IEEE.

Caputo, E., Corallo, A., Damiani, A., & Passiante, G. (2010). KPI Modeling in MDA Perspective. *International conference on On the move to meaningful internet systems* (pp. 384-393). Berlin: Springer.

Constant Field Values. (2014, July). Retrieved from KIE API 6.0.1.Final: http://docs.jboss.org/jbpm/v6.0.1/javadocs/constant-values.html

Daniyal, A., Abidi, S., & Abidi, S. (2009). Computerizing Clinical Pathways: Ontology-Based Modeling and Execution. *22nd International Congress of the European-Federation-for-Medical-Informatics on Medical Informatics Europe (MIE).* Sarajevo, BOSNIA & HERCEG.

Decker, M. (1991). The Development of Indicators. *Infection Control and Hospital Epidemiology*, 490-492.

Deloitte. (2014). *2014 Global health care outlook. Shared challenges, shared opportunities.* Deloitte Touche Tohmatsu Limited.

Donabedian, A. (1988). The quality of care: How can it be assessed? *Journal of the American Medical Association*, 1743-1748.

Du, G., Jiang, Z., & Diao, X. (2008). The integrated modeling and framework of Clinical Pathway Adaptive Workflow Management System based on Extended Workflow-nets (EWF-nets). *IEEE International Conference on Service Operations and Logistics and Informatics, 1&2* (pp. 914-918). Beijing: IEEE.

Du, G., Jiang, Z., Diao, X., Ye, Y., & Yao, Y. (2009). Modelling, Variation Monitoring, Analyzing, Reasoning for Intelligently Reconfigurable Clinical Pathway. *IEEE International Conference on Service Operation, Logistics and Informatics* (pp. 85-90). Chicago: IEEE.

Dumas, M., La Rosa, M., Mendling, J., & Reijers, H. A. (2013). *Fundamentals of Business Process Management.* Springer.

European Pathway Association. (2014, June). Retrieved from http://www.e-p-a.org/

Franceschini, F., Galetto, M., & Maisano, D. (2007). *Management by Measurement: Designing Key Indicators and Performance Measurement Systems.* Berlin: Springer.

H2 Database. (2014, July). Retrieved from H2 Database: http://www.h2database.com/html/faq.html#reliable

Harmon, P. (2007). *Exploring BPMS with Free or Open Source Products.* Retrieved from BPTrends: Business Process Trencds: http://www.bptrends.com/publicationfiles/advisor200707311.pdf

Haspeslagh, M., Vanhaecht, K., De Witte, K., Sermeus, W., van de Waeter, W., & Serra, F. (2002). Ontwikkelen en testen van een instrument voor het meten van teameffectiviteit in het kader van klinische paden. *Acta Hospitalia*, 117-122.

Hornix, P. (2007). *Performance Analysis of Business Processes through Process Mining (Master's Thesis).* Eindhoven: Eindhoven University of Technology.

Hu, Z., Li, J.-s., Yu, H.-y., Zhang, X.-g., Suzuki, M., & Araki, K. (2009). Modeling of Clinical Pathways Based on Ontology. *2009 IEEE INTERNATIONAL SYMPOSIUM ON IT IN MEDICINE & EDUCATION* (pp. 1170-1174). Jinan, PEOPLES R CHINA: IEEE.

Hyett, K., Podosky, M., Santamaria, N., & Ham, J. (2007). Valuing variance: the importance of variance analysis in clinical pathways utilisation. *AUSTRALIAN HEALTH REVIEW vol 31 issue 4*, 565-570.

IBM. (2014, July). *IBM.* Retrieved from SQL Guide: http://publib.boulder.ibm.com/infocenter/soliddb/v6r3/index.jsp?topic=/com.ibm.swg.im.soliddb.sql.doc/doc/tables.rows.and.columns.html

Interface ProcessInstance. (2014, July). Retrieved from KIE API 6.0.1.Final: http://docs.jboss.org/jbpm/v6.0.1/javadocs/org/kie/api/runtime/process/ProcessInstance.html

ISO. (2011). *ISO/IEC 25010:2011 Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models.*

iso25000.com. (2014, October). Retrieved from ISO 25000 Software Product Quality: http://iso25000.com/index.php/en/iso-25000-standards/iso-25010

JBoss Developer Forum. (2014, July). Retrieved from JBoss Developer Forum: https://community.jboss.org/thread/242050

jBPM - JBoss Community. (2014, March). Retrieved from jBPM - JBoss Community: http://jbpm.jboss.org/

jBPM Human Tasks. (2014, March). Retrieved from jBPM 6.0.1 Documentation: http://docs.jboss.org/jbpm/v6.0.1/userguide/jBPMTaskService.html

jBPM Installer. (2014, May). Retrieved from jBPM Documentation: http://docs.jboss.org/jbpm/v6.1/userguide/jBPMInstaller.html

jBPM Overview. (2014, March). Retrieved from jBPM 6.0.1 Documentation: http://docs.jboss.org/jbpm/v6.0.1/userguide/jBPMOverview.html

jBPM Persistence and Transactions. (2014, April). Retrieved from jBPM 6.0.1 Documentation: http://docs.jboss.org/jbpm/v6.0.1/userguide/jBPMPersistence.html

jBPM Processes. (2014, March). Retrieved from jBPM 6.0.1 Documentation: http://docs.jboss.org/jbpm/v6.0.1/userguide/jBPMBPMN2.html

Lebas, M. (1995). Performance measurement and performance management. *International Journal of Production Economics*, 23-35.

Lerner, B., Christov, S., Osterweil, L., Bendraou, R., Kannengiesser, U., & Wise, A. (2010). Exception Handling Patterns for Process Modeling. *IEEE Transactions on Software Engineering vol 36 issue 2*, 162-183.

Mainz, J. (2003). Defining and classifying clinical indicators for quality improvement. *International Journal for Quality in Health Care*, vol 15 no 6 pp 523-530.

March, J., & Sutton, R. (1997). Organizational Performance as a Dependent Variable. *ORGANIZATION SCIENCE vol 8 issue 6*, 698-706.

Mathwords. (2014, August). Retrieved from Mathwords: Expression: http://www.mathwords.com/e/expression.htm

Meenakshy, P. (2013). *A Performance Measurement Framework for Clinical Pathways Monitoring (Master's Thesis).* Eindhoven: Eindhoven University of Technology.

Mernik, M., Heering, J., & Sloane, A. (2005). When and How to Develop Domain-Specific Languages. *ACM Computing Surveys, Vol. 37, No. 4*, 316–344.

Merriam Webster Dictionary. (2014, June). Retrieved from Merriam Webster Dictionary: http://www.merriam-webster.com/dictionary/performance

Object Management Group. (2011). *Business Process Model and Notation (BPMN) Version2.0.* Object Management Group.

Object Management Group. (2014, 07). Retrieved from BPMN Specification: http://www.bpmn.org

planet.jboss.org. (2014, July). Retrieved from jBPM 6 - store your process variables anywhere: http://planet.jboss.org/post/jbpm_6_store_your_process_variables_anywhere

PostgreSQL Connections and Authentication. (2014, July). Retrieved from PostgreSQL: http://www.postgresql.org/docs/9.3/static/runtime-config-connection.html#GUC-LISTEN-ADDRESSES

PostgreSQL pg_hba.conf. (2014, July). Retrieved from PostgreSQL: http://www.postgresql.org/docs/9.3/static/auth-pg-hba-conf.html

Schrijvers, G., van Hoorn, A., & Huiskes, N. (2012). The care pathway: concepts and theories: an introduction. *International Journal of Integrated Care*, Vol 12.

Silberschatz, A., Korth, H. F., & Sudarshan, S. (2011). *Database System Concepts, Sixth Edition.* New York: McGraw-Hill.

Taylor, A. (2010). *SQL for Dummies 7th edition.* Indianapolis: Wiley Publishing.

The JBoss Drools team. (2013). *Drools Documentation Version 6.0.1.Final.*

van Gerven, E., Vanhaecht, K., Deneckere, S., Vleugels, A., & Sermeus, W. (2010). Management challenges in care pathways: conclusions of a qualitative study within 57 health-care organizations. *International Journal of Care Pathways*, 142-149.

van Renswouw, W. (2013). *From paper-based care pathway to executable workflow process model (Master's Thesis).* Eindhoven: Eindhoven University of Technology.

Vanhaecht, K., & Sermeus, W. (2003). The Leuven Clinical Pathway Compass. *Journal of Integrated Care Pathways*, 2-7.

Vanhaecht, K., Panella, M., van Zelm, R., & Sermeus, W. (2010). An overview on the history and concept of care pathways as complex interventions. *International Journal of Care Pathways*, 117-123.

Vanhaecht, K., van Gerven, E., Deneckere, S., Lodewijckx, C., Janssen, I., van Zelm, R., . . . Sermeus, W. (2012). The 7-phase method to design, implement and evaluate care pathways. *The International Journal of Person Centered Medicine Vol 2 Issue 3*, 341-351.

Vanhaecht, K.; De Witte, K.; Sermeus, W. (2007). *The impact of clinical pathways on the organisation of care processes.* Leuven: ACCO.

Vermeulen, L. (2013). *A Process Modeling Method for Care Pathways (Master's Thesis).* Eindhoven: Eindhoven University of Technology.

Volzer, H. (2010). An Overview of BPMN 2.0 and Its Potential Use. *2nd International Workshop on Business Process Modeling Notation.* Potsdam: Springer-Verlag Berlin, Germany.

Wakamiya, S., & Yamauchi, K. (2006). A new approach to systematization of the management of paper-based clinical pathways. *Computer Methods and Programs in Biomedicine*, 169-176.

Wakamiya, S., & Yamauchi, K. (2009). What are the standard functions of electronic clinical pathways? *International Journal of Medical Informatics*, 543-550.

Walker, R. D., Howard, M. O., Lambert, M. D., & Suchinsky, R. (1994). Medical practice guidelines. *West J Med*, 39-44.

Wohed, P., Russel, N., ter Hofstede, A., Andersson, B., & van der Aalst, W. (2008). Open Source Workflow: A Viable Direction for BPM? Extended Abstract. *20th International Conference on Advanced Information Systems Engineering* (pp. 583-586). Montpellier: Springer-Verlag Berlin.

Ye, Y., Jiang, Z., Diao, X., Yang, D., & Du, G. (2009). An ontology-based hierarchical semantic modeling approach to clinical pathway workflows. *Computers in Biology and Medicine*, 722-732.

APPENDIX A List of BPMN elements supported by jBPM 6

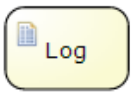
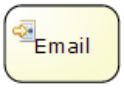
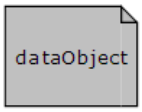
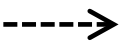
A list of BPMN elements supported by jBPM 6.0.1 and their description (Object Management Group, 2011) can be seen in Table 14. The list and the symbols are obtained from KIE Workbench's process designer.

Table 14 BPMN elements supported by jBPM 6

Task	
	**User** *A User Task is a Task which is performed by human with the assistance of a software application and is scheduled through a task list manager.*
	**Service** *A Service Task is a Task that uses some sort of service, which could be a Web service or an automated application.*
	**Send** *A Send Task is a Task that is designed to send a Message to an external Participant (relative to the Process). Once the Message has been sent, the Task is completed.*
	**Business Rule** *A Business Rule Task provides a mechanism for the Process to provide input to a Business Rules Engine and to get the output of calculations that the Business Rules Engine might provide.*
	**Receive** *A Receive Task is a simple Task that is designed to wait for a Message to arrive from an external Participant (relative to the Process). Once the Message has been received, the Task is completed.*
	**Script** *A Script Task is executed by a business process engine. The modeler or implementer defines a script in a language that the engine can interpret. When the Task is ready to start, the engine will execute the script. When the script is completed, the Task will also be completed.*
	**Manual** *A Manual Task is a Task that is expected to be performed without the aid of any business process execution engine or any application.*
Subprocess	
	**Reusable** *This subprocess calls a pre-defined Process.*
	**Multiple Instance** *This subprocess allows for creation of a desired number of Activity instances. The instances may execute in parallel or may be sequential.*
	**Embedded** *A nested (or embedded) Sub-Process is an activity that shares the same set of data as its parent process*

	**Ad-Hoc** *An Ad-Hoc Sub-Process is a specialized type of Sub-Process that is a group of Activities that have no REQUIRED sequence relationships.*
	**Event** *An Event Sub-Process is a specialized Sub-Process that is used within a Process (or Sub-Process). An Event Sub-Process is not part of the normal flow of its parent Process—there are no incoming or outgoing Sequence Flows.*
Start Event	
	**None** *The None Start Event does not have a defined trigger.*
	**Conditional** *This type of event is triggered when a condition such as "S&P 500 changes by more than 10% since opening", or "Temperature above 300C" become true.*
	**Message** *A Message arrives from a Participant and triggers the start of the Process.*
	**Timer** *A specific time-date or a specific cycle (e.g., every Monday at 9am) can be set that will trigger the start of the Process.*
	**Escalation** *Escalation Event Sub-Processes implement measures to expedite the completion of a business Activity, should it not satisfy a constraint specified on its execution (such as a time-based deadline). The Escalation Start Event is only allowed for triggering an in-line Event Sub- Process.*
	**Error** *The Error Start Event is only allowed for triggering an in-line Event Sub- Process.*
	**Compensation** *The Compensation Start Event is only allowed for triggering an in-line Compensation Event Sub-Process. This type of Event is triggered when compensation occurs.*
	**Signal** *A Signal arrives that has been broadcast from another Process and triggers the start of the Process.*
End Event	
	**None** *The None End Event does not have a defined result.*
	**Cancel** *This type of End indicates that the Transaction should be cancelled and will trigger a Cancel Intermediate Event attached to the Sub-Process boundary.*
	**Message** *This type of End indicates that a Message is sent to a Participant at the conclusion of the Process.*
	**Escalation** *This type of End indicates that an Escalation should be triggered. Other active threads are not affected by this and continue to be executed.*
	**Error** *This type of End indicates that a named Error should be generated. All currently active threads in the particular Sub-Process are terminated as a result.*
	**Compensation** *This type of End indicates that compensation is necessary. If an Activity is identified, and it was successfully completed, then that Activity will be compensated.*
	**Signal** *This type of End indicates that a Signal will be broadcasted when the End has been reached.*

	**Terminate** *This type of End indicates that all Activities in the Process should be immediately ended. This includes all instances of multi-instances. The Process is ended without compensation or event handling.*
Catching Intermediate Event	
	**Message** This event is triggered when a Message is received.
	**Timer** *In normal flow the Timer Intermediate Event acts as a delay mechanism based on a specific time-date or a specific cycle (e.g., every Monday at 9am) can be set that will trigger the Event.*
	**Escalation** This event reacts on the escalation. It needs to be attached to the boundary of an activity.
	**Conditional** *This type of Event is triggered when a condition becomes true.*
	**Error** *A catch Intermediate Error Event can only be attached to the boundary of an Activity*
	**Compensation** Compensation handling in case of partially failed operation.
	**Signal** *This type of Event is used for receiving Signals*
Throwing Intermediate Event	
	**Message** *This event will send a Message.*
	**Escalation** *This event will raise an Escalation.*
	**Signal** *This type of event is used for sending Signals*
Gateway	
	**Data-based Exclusive (XOR)** *A diverging Exclusive Gateway (Decision) is used to create alternative paths within a Process flow, only one of the paths can be taken.* *A converging Exclusive Gateway is used to merge alternative paths*
	**Event-based** *The Event-Based Gateway represents a branching point in the Process where the alternative paths that follow the Gateway are based on Events that occur, rather than the evaluation of Expressions using Process data (as with an Exclusive or Inclusive Gateway).*
	**Parallel** *A Parallel Gateway is used to synchronize (combine) parallel flows and to create parallel flows.*
	**Inclusive** *A diverging Inclusive Gateway (Inclusive Decision) can be used to create alternative but also parallel paths within a Process flow. Unlike the Exclusive Gateway, all condition Expressions are evaluated. The true evaluation of one condition Expression does not exclude the evaluation of other condition Expressions.. However, it should be designed so that at least one path is taken. A converging Inclusive Gateway is used to merge a combination of alternative and parallel paths*

Service Task	
	**Log** A jBPM specific service task
	**Email** A jBPM specific service task
Data Object	
	**Data Object** *Data Objects provide information about what Activities require to be performed and/or what they produce*
Swimlane	
	**Lane** *Lanes are used to organize and categorize Activities.*
Artifact	
	**Group** *A Group is a grouping of graphical elements that are within the same Category. This type of grouping does not affect the Sequence Flows within the Group. Categories can be used for documentation or analysis purposes. Groups are one way in which Categories of objects can be visually displayed on the diagram.*
	**Text Annotation** *Text Annotations are a mechanism for a modeler to provide additional text information for the reader of a BPMN Diagram*
Connecting Object	
	**Sequence Flow** *A Sequence Flow is used to show the order that Activities will be performed in a Process and in a Choreography.*
	Association (un-directed) *An Association is used to link information and Artifacts with BPMN graphical elements. Text Annotations and other Artifacts can be Associated with the graphical elements.*
	**Association (uni-directional)** *An arrowhead on the Association indicates a direction of flow, when appropriate.*

APPENDIX B Persisting jBPM 6.1.0.CR1 historical data in PostgreSQL 9.3

To persist jBPM historical data in PostgreSQL, the steps below are followed. This instruction is derived from jBPM documentation (jBPM Installer, 2014). Note that this instruction does not cover all installation steps for jBPM, i.e. Java and Ant settings. Full instruction can be found in the documentation (jBPM Installer, 2014).

1. Create a username and password in PostgreSQL, e.g. username=jbpm, password=jbpm. Note that the username should be able to create database and create table.
2. Using username created in step #1, create a database, for example *jbpm*

3. In jBPM installation folder (will be called <*jbpm_home*> from now), open *build.properties*. Comment the following lines to disable H2:

```
# H2.version=1.3.170

# db.name=h2

# db.driver.jar.name=${db.name}.jar

# db.driver.download.url=http://repo1.maven.org/maven2/com/

h2database/h2/${H2.version}/h2-${H2.version}.jar
```

   And un-comment the following lines to enable PostgreSQL:

```
db.name=postgresql

db.driver.module.prefix=org/postgresql

db.driver.jar.name=${db.name}-jdbc.jar

db.driver.download.url=https://repository.jboss.org/nexus/content/repositories
/thirdparty-uploads/postgresql/postgresql/9.1-902.jdbc4/postgresql-9.1-
902.jdbc4.jar
```

   The script above will install PostreSQL jdbc driver as JBoss module.

4. Open *<jbpm_home>/db/jbpm-persistence-JPA2.xml*. Change the value for property *hibernate.dialect* to PostgreSQL dialect as follows.

```
<property name="hibernate.dialect"
value="org.hibernate.dialect.PostgreSQLDialect" />
```

5. Create a file and give it name *postgresql-module.xml* and put it in *<jbpm_home>/db/*. Write the following in the file.

```
<module xmlns="urn:jboss:module:1.0" name="org.postgresql">
   <resources>
     <resource-root path="postgresql-jdbc.jar"/>
```

```
   </resources>
   <dependencies>
      <module name="javax.api"/>
      <module name="javax.transaction.api"/>
    </dependencies>
</module>
```

6. In *<jbpm_home>* , open all standalone-*.xml files. Find *<datasources>* element. Replace the entries inside *<datasources>* and *</datasources>* with the following.

```
<datasource jndi-name="java:jboss/datasources/jbpmDS" pool-
name="PostgreSQLDS" enabled="true" use-java-context="true">
  <connection-url>jdbc:postgresql://localhost:5432/jbpm</connection-url>
  <driver>postgresql</driver>
  <pool></pool>
  <security>
    <user-name>jbpm</user-name>
    <password>jbpm</password>
  </security>
</datasource>
<drivers>
  <driver name="postgresql" module="org.postgresql">
    <xa-datasource-class>org.postgresql.xa.PGXADataSource</xa-datasource-
class>
  </driver>
</drivers>
```

`<connection-url>` specifies the address of database being used. In the example above the database name is *jbpm* and it is located in *localhost:5432*. If you want to use different database url/name, username and password, change them in `<connection-url>`,`<user-name>` and `<password>` respectively. This should be similar to what was created in step #1 and #2.

7. If you already install jBPM (with the same <jbpm_home>) prior to modifying files above, it is recommended that you reinstall jBPM with the following commands in the given order.

`ant stop.demo`	(to stop jBPM, if it is running)
`ant clean.demo`	(to uninstall jBPM)
`ant install.demo.noeclipse`	(to install jBPM without Eclipse IDE extension
`ant start.demo.noeclipse`	(to start jBPM demo without Eclipse IDE)

8. To check whether JDBC connection to database can be made successfully, open jBPM administrator console in *http://localhost:8080/console*, It will be redirected to another page (http://localhost:9990/console/App.html) and ask for username and password for administrator. If you have not created administrator account, create one by executing <jbpm-home>/jboss-as-7.1.1.Final/bin/add-user.bat. For example can be seen in Figure 27.

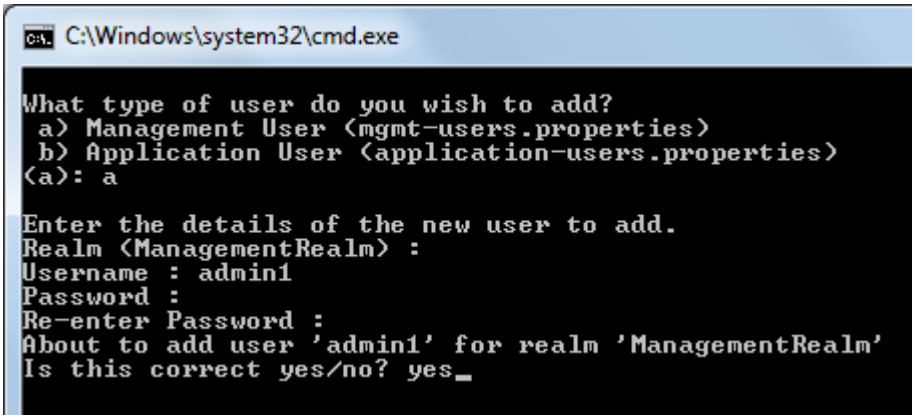


Figure 27 Create jBPM management user

In administrator console, go to Profile (top right) -> Connector -> Datasources. In the JDBC Datasources page, click on the data source name we created using this instruction, i.e. `PostgreSQLDS` with JNDI name `java:jboss/datasources/jbpmDS`(see #6). Click on tab *Connection*. Click button *Test Connection.* See the message if JDBC connection is successful.

9. If the connection is successful, open the default address for jBPM demo: *http://localhost:8080/jbpm-console* and login using default username & password, e.g. username: krisv password: krisv. After you log in, the tables required will be created automatically in the database. Check the created tables using PostgreSQL pgAdmin

10. Some possible errors that might occur are: not able to make JDBC connection or cannot start jBPM (failed deployment, indicated by ".failed" file(s) is created in *<jbpm_home>/jboss-as-7.1.1.Final/standalone/deployments*). To solve this problem, try these steps.

    a. Check standalone-*.xml files, whether the entries, especially in data source section are correct. Single unnecessary character will cause error.

    b. Check if your PostgreSQL service is running

    c. Check whether you have created database as specified in standalone-*.xml files

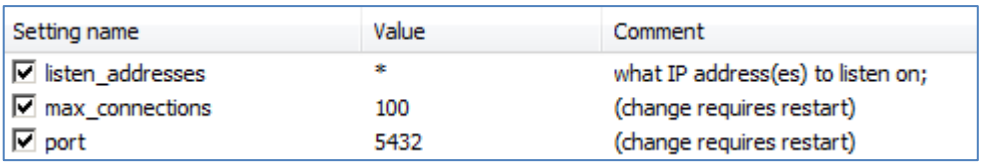
    d. Check whether PostgreSQL is set to allow TCP/IP remote connections. Default installataion of PostgreSQL does not allow remote connection. To allow PostgreSQL for remote connections, do the following.
    Open pg_hba.conf (File -> Open pg_hba.conf), using PostgreSQL pgAdmin. The file is located in *<PostgreSQL install folder>/data*. You can also modify the file directly using text editor. Add the entries for example as shown in Figure 28.

Type	Database	User	IP-Address	Method	Option
☑ host	all	all	127.0.0.1/32	md5	
☑ host	all	all	::1/128	md5	
☑ host	all	all	127.0.0.1/32	md5	
☑ host	all	all	::1/128	md5	

Figure 28 PostgreSQL pg_hba.conf

The first line of the entries means that PostgreeSQL allows that the connection attempt is made using TCP/IP, for all database, for all users, from client address 127.0.0.1 (also known as localhost). 32 is the CIDR mask length for IP mask 255.255.255.255. Complete options for this setting can be found in (PostgreSQL pg_hba.conf, 2014).

Remote TCP/IP connections are only possible with appropriate value for *listen_adresses* configuration parameter. You can set it in *postgresql.conf* file (similar step with pg_hba.conf). In this master thesis we set the value to * which means that the server listens for connections from client applications in all available IP interfaces (see Figure 29). Complete options for this parameter can be found in (PostgreSQL Connections and Authentication, 2014). Restart your PostgreSQL service after doing all modifications above.



Setting name	Value	Comment
☑ listen_addresses	*	what IP address(es) to listen on;
☑ max_connections	100	(change requires restart)
☑ port	5432	(change requires restart)

Figure 29 Setting listen_addresses in postgresql.conf

APPENDIX C Description of jBPM history logs

C.1. ProcessInstanceLog (jBPM Persistence and Transactions, 2014)

Table 15 ProcessInstanceLog table fields definition

Field	Description	Nullable
id	The primary key and id of the log entity	NOT NULL
duration	Actual duration of this process instance since its start date	
end_date	When applicable, the end date of the process instance	
externalId	Optional external identifier used to correlate to some elements - e.g. deployment id	
user_identity	Optional identifier of the user who started the process instance	
outcome	The outcome of the process instance, for instance error code in case of process instance was finished with error event	
parentProcessInstanceId	The process instance id of the parent process instance if any	
processid	The id of the process	
processinstanceid	The process instance id	NOT NULL
processname	The name of the process	
processversion	The version of the process	
start_date	The start date of the process instance	
status	The status of process instance that maps to process instance state	

C.2. NodeInstanceLog (jBPM Persistence and Transactions, 2014)

Table 16 NodeInstanceLog table fields definition

Field	Description	Nullable
id	The primary key and id of the log entity	NOT NULL
connection	Actual identifier of the sequence flow that led to this node instance	
log_date	The date of the event	
externalId	Optional external identifier used to correlate to	

	some elements - e.g. deployment id	
nodeid	The node id of the corresponding node in the process definition	
nodeinstanceid	The node instance id	
nodename	The name of the node	
nodetype	The type of the node	
processid	The id of the process that the process instance is executing	
processinstanceid	The process instance id	NOT NULL
type	The type of the event (0 = enter, 1 = exit)	NOT NULL
workItemId	Optional - only for certain node types - The identifier of work item	

C.3. VariableInstanceLog (jBPM Persistence and Transactions, 2014)

Table 17 VariableInstanceLog table fields definition

Field	Description	Nullable
id	The primary key and id of the log entity	NOT NULL
externalId	Optional external identifier used to correlate to some elements - e.g. deployment id	
log_date	The date of the event	
processid	The id of the process that the process instance is executing	
processinstanceid	The process instance id	NOT NULL
oldvalue	The previous value of the variable at the time that the log is made	
value	The value of the variable at the time that the log is made	
variableid	The variable id in the process definition	
variableinstanceid	The id of the variable instance	

APPENDIX D jBPM Experiment

D.1. Test Model 1

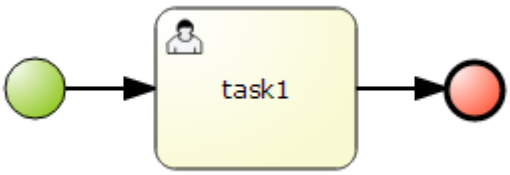


Figure 30 Test model 1

D.2. Test model 2

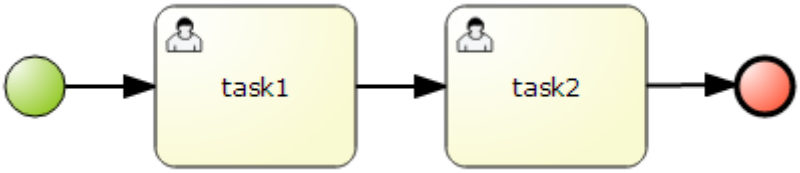


Figure 31 Test model 2

D.3. Test model 3

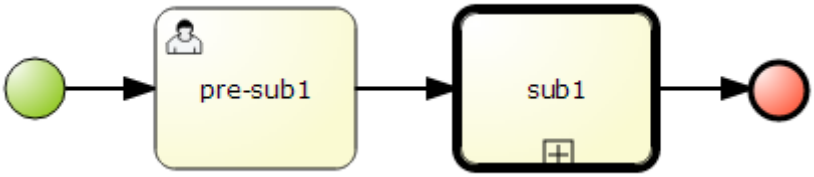


Figure 32 Test model 3

D.4. Execution Result

ProcessInstanceLog

id bigi	start_date timestamp without time zone	end_date timestamp without time zone	user_identity character varying(255)	processid character varying(255)	processname character varying(255)	processinstanceid bigint	status integer
10	2014-07-27 16:20:20.123	2014-07-27 16:20:59.706	krisv	masterthesis.relati	relationshiptest	10	2

NodeInstanceLog

id bigint	connection character va	log_date timestamp w	externalid character va	nodeid characte	nodeinstanceid character varyin	nodename character va	nodetype character varyin	processid character v	processinstanceid bigint	type integer	workitemid bigint
66		2014-07-27	org.jbpm.ma	process	0		StartNode	masterthes	10	0	
67	54AAF340-1	2014-07-27	org.jbpm.ma	C1DC7F2	1	task1	HumanTaskNode	masterthes	10	0	
68	54AAF340-1	2014-07-27	org.jbpm.ma	process	0		StartNode	masterthes	10	1	
69	F27945A4-1	2014-07-27	org.jbpm.ma	B317E5(	2		EndNode	masterthes	10	0	
70		2014-07-27	org.jbpm.ma	B317E5(	2		EndNode	masterthes	10	1	
71	F27945A4-1	2014-07-27	org.jbpm.ma	C1DC7F2	1	task1	HumanTaskNode	masterthes	10	1	15

VariableInstanceLog

id bigi	log_date timestamp	externalid character v	oldvalue character	processid character v	processinstanceid bigint	value charact	variableid character va	variableinstanceid character varying(255)
16	2014-07-2	org.jbpm.r		masterthes	10	2	var1	var1

BAMTaskSummary

pk big	createddate timestamp without time zone	duration bigint	enddate timestamp without time zone	processinstanceid bigint	startdate timestamp without time zone	status character va	taskid bigint	taskname character v	userid charact	optlock integer
15	2014-07-27 16:20:20.145	22690	2014-07-27 16:20:59.772	10	2014-07-27 16:20:37.082	Completed	15	task1	krisv	2

AuditTaskImpl

id bigi	activationtime date	actualowner character var	createdby character v	createdon date	deploymentid character varyi	description character va	duedate date	name charact	parentid bigint	priority integer	processid character v	processinstanceid bigint	processsessionid integer	status character v	taskid bigint
15	2014-07-27	krisv	krisv	2014-07-27	org.jbpm.mast			task1	-1	0	masterthes	10	6	Completed	15

TaskEvent

id bigint	logtime timestamp without time zone	taskid bigint	type character varying(255)	userid character varying(255)	optlock integer
53	2014-07-27 16:20:20.148	15	ADDED		0
54	2014-07-27 16:20:37.082	15	STARTED	krisv	0
55	2014-07-27 16:20:59.772	15	COMPLETED	krisv	0

Figure 33 Execution result of scenario 1

VariableInstanceLog

id bigi	log_date timestamp	externalid character v	oldvalue character	processid character v	processinstanceid bigint	value character v	variableid character va	variableinstanceid character varying(255)
17	2014-07-3	org.jbpm.r		masterthes	11	masterthe	processId	processId
18	2014-07-3	org.jbpm.r		masterthes	11	5	var1	var1
19	2014-07-3	org.jbpm.r	5	masterthes	11	6	var1	var1
20	2014-07-3	org.jbpm.r	6	masterthes	11	7	var1	var1

Figure 34 Execution result of scenario 2

BAMTaskSummary

pk big	createddate timestamp without time zone	duration bigint	enddate timestamp without time zone	processinstanceid bigint	startdate timestamp without time zone	status character va	taskid bigint	taskname character v	userid charact	optlock integer
18	2014-07-31 11:35:25.366	14287	2014-07-31 11:39:11.236	12	2014-07-31 11:38:56.949	Completed	18	task1	krisv	5

AuditTaskImpl

id bigi	activationtime date	actualowner character var	createdby character v	createdon date	deploymentid character varyi	description character va	duedate date	name charact	parentid bigint	priority integer	processid character v	processinstanceid bigint	processsessionid integer	status character v	taskid bigint
18	2014-07-31	krisv	krisv	2014-07-31	org.jbpm.mast			task1	-1	0	masterthes	12	9	Completed	18

TaskEvent

id bigint	logtime timestamp without time zone	taskid bigint	type character varying(255)	userid character varying(255)	optlock integer
62	2014-07-31 11:35:25.363	18	ADDED		0
63	2014-07-31 11:35:42.318	18	STARTED	krisv	0
64	2014-07-31 11:38:51.257	18	RELEASED		0
65	2014-07-31 11:38:56.902	18	CLAIMED	krisv	0
66	2014-07-31 11:38:56.942	18	STARTED	krisv	0
67	2014-07-31 11:39:11.228	18	COMPLETED	krisv	0

Figure 35 Execution result of scenario 3

ProcessInstanceLog

id big	duration bigint	end_date timestamp without time zone	externalid character v	user_identity character var	outcome character	parentprocessinstanceid bigint	processid character v	processinstanceid bigint	processname character vary	processversion character varyir	start_date timestamp v	status integer
13	29111	2014-07-31 15:22:13.555	org.jbpm.:	krisv			masterthe	13	subprocesste	1.0	2014-07-31	2
14	16898	2014-07-31 15:22:13.468	org.jbpm.:	krisv		13	masterthe	14	relationship	1.0	2014-07-31	2

Figure 36 Execution result of scenario 4

APPENDIX E Exception handling patterns

Table 18 Exception handling patterns

Pattern	Description
Immediate Fixing	When a nonnormative situation is noted, an action is taken to address the problem that caused this situation before continuing with the remainder of the process. This is done by representing the nonnormative situation as an Intermediate (catch) Event attached to the boundary of a task. When an Event with one of the specified triggers is detected, the flow will be redirected to the Intermediate Event and continue with fixing activity(called Exception Flow) before rejoining the normative path (Lerner, et al., 2010). The structure of this pattern is shown in Figure 37.
Deferred Fixing	When a nonnormative situation is noted, action must be taken to record the situation and possibly address the situation either partially or temporarily because full resolution is either not immediately possible or not necessary. Similar to immediate fixing, this is implemented by representing the nonnormative situation as an Intermediate (catch) Event attached to the boundary of a task. The exception flow will execute temporary action and record the problem. Later in the future of flow, after rejoining the normative path, a gateway is used to check if full fixing is required (Lerner, et al., 2010). The structure of this pattern is shown in Figure 39 .
Retry	When a problem is detected immediately after the execution of the activity causing the problem, an action is taken to address the problem (in the exception flow) and then the activity that caused the problem is tried again. This is done by representing the nonnormative situation as an Intermediate (catch) Event attached to the boundary of a task. If the problem is not resolved, the fixing action is repeated until the problem is resolved or retry is no longer possible (Lerner, et al., 2010). The structure of this pattern is shown in Figure 38.
Reject	This exception occurs when the process cannot continue and hence it should end. This is done by representing the nonnormative situation as an Intermediate (catch) Event attached to the boundary of a task. When an Event with one of the specified triggers is detected, the flow will be redirected to the Intermediate Event and continue with an End Event, for example a message event (Lerner, et al., 2010). The structure of this pattern is shown in Figure 40.
Compensate	This pattern is intended to address the need to determine what work must be undone when cancelling an activity and determine what action should be performed to compensate that undone work. In BPMN, it is done by using Compensation construct which rolls back some of the effect of a Transaction. A

	Transaction is based on a formal business relationship and unanimous agreement among two or more participants and is modeled as a Sub-Process. A Cancellation Event is attached to the Sub-Process. When this Cancellation Event is triggered, all completed activity in the Sub-Process with Compensation Event attached is rolled back by executing defined Cancellation Task (Lerner, et al., 2010). The structure of this pattern is shown in Figure 41.

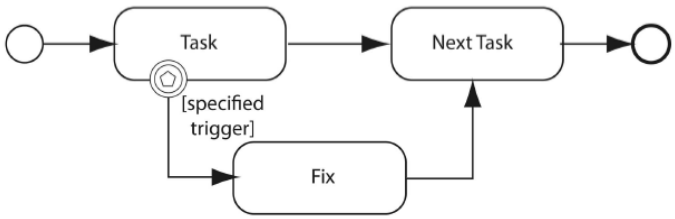


Figure 37 Immediate fixing pattern

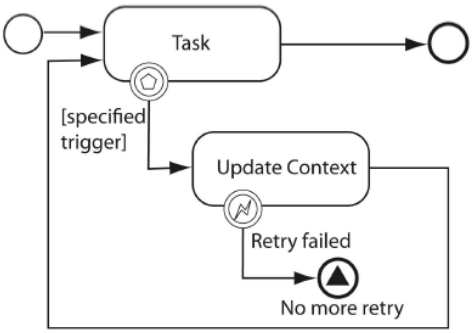


Figure 38 Retry pattern

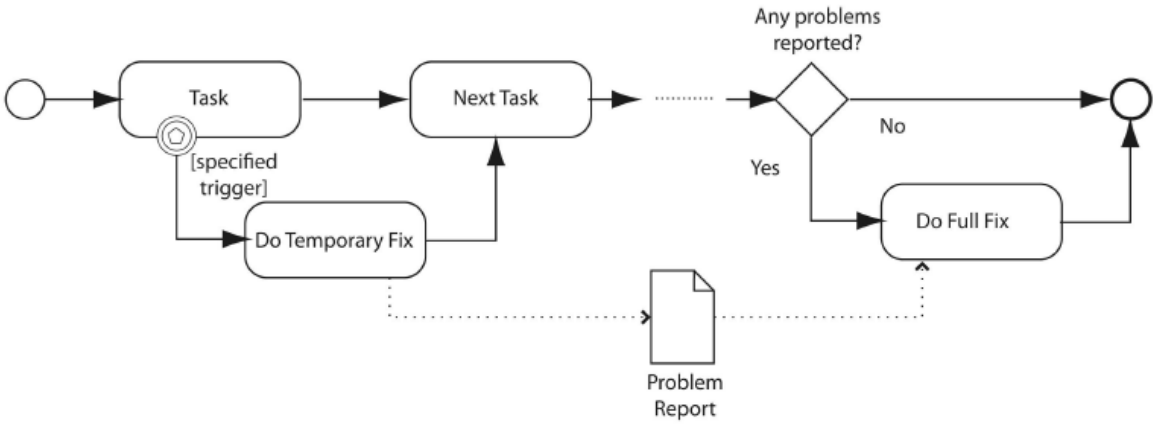


Figure 39 Deferred fixing pattern

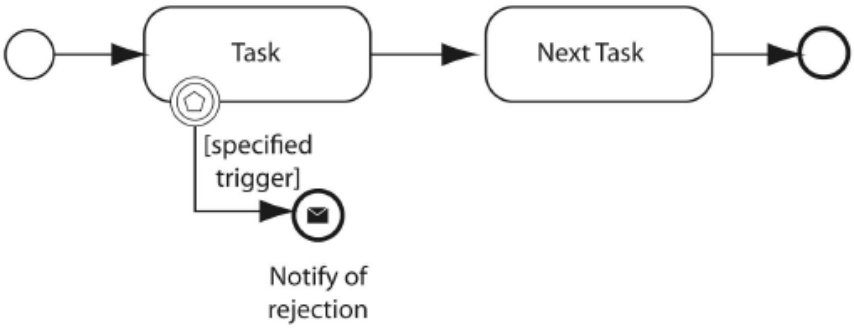


Figure 40 Reject pattern

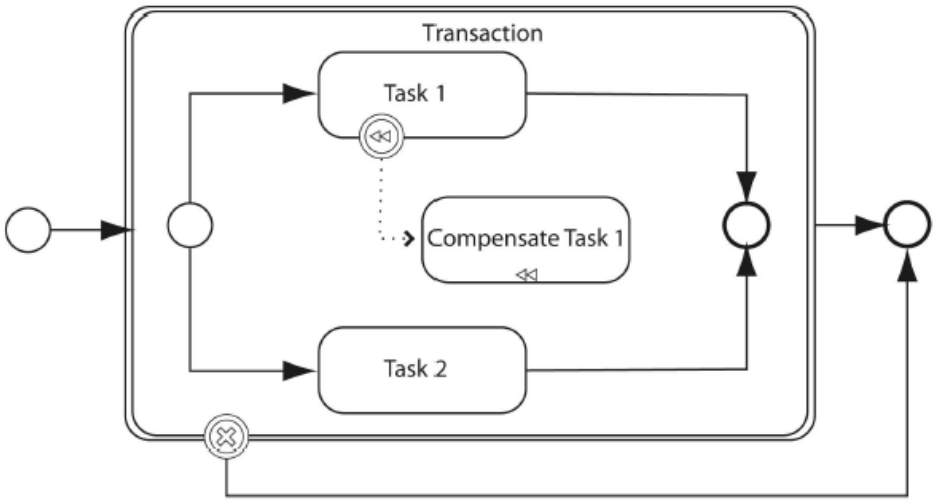


Figure 41 Compensate pattern

APPENDIX F Formulation of Unstable Angina Pathway KPI

The formulation of UA KPI (Vermeulen, 2013) in conjunction with eight care pathway measures identified is given in Table 19.

Table 19 Formulation of Unstable Angina Pathway KPI

Domain	Performance indicator	Description	Formulation
Clinical	Patients admitted to the CCU or EHH	Percentage of UA patients that are admitted to the CCU or EHH	Number of patient (#1) with data element condition "admitted" = true divided by Number of patient (#1).
Clinical	Troponin measuring (%)	Percentage of UA patients, where Troponin is measured	Number of patient (#1) with data element condition "troponin_measured" = true divided by Number of patient (#1)
Clinical	Use of aspirin	Percentage UA patients where aspirin is prescribed during hospitalization	Number of patient (#1) with data element condition "aspirin_prescribed" = true divided by Number of patient (#1)
Clinical	Use of tricagrelor or clopidogrel	Percentage UA patients where tricagrelor or clopidogrel is prescribed during hospitalization	(Number of patient (#1) with data element condition "tricagrelor_prescribed" = true + Number of patient (#1) with data element condition "clopodogrel_prescribed" = true) divided by Number of patient (#1)
Clinical	Use of fondaparinux or enoxaparin	Percentage UA patients where fondaparinux or enoxaparin is prescribed during hospitalization	(Number of patient (#1) with data element condition "fondaparinux_prescribed" = true + Number of patient (#1) with data element condition "enoxaparin_prescribed" = true) divided by Number of patient (#1)
Clinical	Use of enoxaparin in patient with kidney failure?	Percentage of UA patients with kidney failure where enoxaparin is prescribed	(Number of patient (#1) with data element condition "kidney_failure" = true AND "enoxaparin_prescribed" = true) divided by Number of patient (#1)
		Percentage of UA patients with kidney failure where Fondaparinux is prescribed	(Number of patient (#1) with data element condition "kidney_failure" = true AND "fondaparinux_prescribed" = true) divided by Number of patient (#1)
Clinical	Use of early invasive procedures by intermediate- to high-risk patients	Percentage UA patients with a GRACE > 108 and/or one or more risk factors that get an CAD within 120 minutes	Number of patient (#1) with data element condition "GRACE" > 100 AND "risk_factor" >= 1 and time difference condition (CAD, pre-CAD) <= 120 minutes divided by Number of patient (#1) with data element condition "GRACE" > 108 AND "risk_factor" >= 1
Clinical	Complications (%)	*numerator: Number of patients in which a re-operation within the same hospitalization is necessary because of a bleeding, with or*	Number of patient (#1) with event condition "re-operation" occurs 1x divided by Number of patient (#1) with data element condition "only_first_operation" = true

Domain	Performance indicator	Description	Formulation
		without a tamponade, graft occlusion or other cardiac cause. denominator: Number of patients undergoing a CABG surgery for the first time.	
Clinical	Percentage of deep sternal wound infections	*numerator: Number of patients who develop a deep sternal wound infection related to muscle, bone and/or mediastinum within 30 days after the operation denominator: Number of patients undergoing a CABG surgery for the first time.*	Number of patients (#1) with data element condition "deep_sternal_wound" = true divided by Number of patients (#1) with data element condition "only_first_operation" = true
Clinical	Percentage CVA with permanent injury	*numerator: Number of patients who develop a postoperative stroke. denominator: Number of patients undergoing a CABG surgery for the first time.*	Number of patients (#1) with data element condition "postoperative_stroke" = true divided by Number of patients (#1) with data element condition "only_first_operation" = true
Clinical	Glycoprotein IIb/IIIa inhibitor given for PCI?		Number of patients (#1) with data element condition "is_pci" = true AND "given_glycoprotein_iib_iiia" = true divided by Number of patients (#1) with data element condition "is_pci" = true
Clinical	Angiographic success (successful PCI <20% stenosis) (%)	numerator: PCI patients with <20% rest stenosis in all lesions where PCI is attempted denominator: Total number of PCI procedures in this hospital.	Number of patients (#1) with data element condition "is_pci" = true AND "rest_stenosis" < 20% divided by Number of patients (#1) with data element condition "is_pci" = true
Clinical	Emergency CABG-operation (%)	Numerator: PCI-patients that underwent an emergency CABG operation after a PCI, during the hospitalization of this PCI procedure. Denominator: Total number of PCI procedures in this hospital.	Number of patients (#1) with data element condition "is_pci" = true AND "CABG" = true divided by Number of patients (#1) with data element condition "is_pci" = true
Clinical	Advice on quitting smoking	Percentage of patients that is given advice to stop smoking	Number of patients (#1) with data element condition "given_advice" = true divided by Number of patients (#1)
Clinical	Golden five medicine prescribed at discharge	Percentage UA patients where the five medicine	Number of patients (#1) with data element condition "five_medicine_prescribed" = true

Domain	Performance indicator	Description	Formulation
		ASA, thienopyridine, statin, beta blocker and ACE inhibitor (or ATII) are prescribed at discharge	divided by Number of patients (#1)
Clinical	Beta-blocker at discharge by patients with LV dysfunction	Percentage UA patients with LV dysfunction where the beta blocker is prescribed at discharge	Number of patients (#1) with data element condition "LV dysfunction" = true AND "beta blocker" = true divided by Number of patients(#1) with data element condition "LV dysfunction" = true
Clinical	Use of Statins	Percentage UA patients where statins is prescribed at discharge	Number of patients (#1) with data element condition "statins_prescribed" = true divided by Number of patients (#1)
Clinical	Use of ACE-inhibitor of ARB	Percentage UA patients where Ace-inhibitor or ARB is prescribed at discharge	Number of patients (#1) with data element condition "Ace-inhibitor or ARB" = true divided by Number of patients (#1)
Clinical	Antacid prescribed for patient with gastric disorder		Number of patients (#1) with data element "gastric_disorder" = true AND "antacid_prescribed" = true divided by Number of patients (#1) with data element "gastric_disorder" = true
Clinical	Sign up for Heart rehabilitation	Percentage UA patients that is signed up for heart rehabilitation at discharge	Number of patients (#1) with data element "heart_rehabilitation" = true divided by Number of patients (#1)
Clinical	Sign up for X-ergometry	Percentage UA patients that is signed up for an X-ergometry at discharge	Number of patients (#1) with data element "x-ergometry" = true divided by Number of patients (#1)
Clinical	Major bleeds	Percentage UA patients that have major bleedings during hospitalization	Number of patients (#1) with data element "major_bleedings" = true divided by Number of patients (#1)
Service	Patient satisfaction about pathway		The analysis depends on the form of the questionnaire. If single question, i.e. 10 scale rating is used, the indicator can be formulatied as: Average of Data elements value (#4) for data element name = "satisfaction_rate"
Team	Trained HCK team		Cannot be formulated in process model context. However it can be calculated by counting the number of HCK team members who come to training.
Team	Equal contribution from HCK team members		Cannot be formulated using one of eight measures. If it is defined as "fair job distribution" within role, there is a possibility to calculate it, e.g. by comparing total working time or number of instance per actor or number of task executed. However, many factors needs to be considered for further analysis, e.g. the variation of skill and

Domain	Performance indicator	Description	Formulation
			competence, the policy in the hospital, scheduling, etc.
Team	Effectiveness of team	The effectiveness of a multidisciplinary team based on Fry's theory of focusing on shared goals, clear role definitions, clear procedures and, finally good team relationships, also noted as the Leuven Team effectiveness Scale.	Cannot be formulated using one of eight measures.
Process	ECG done within 10 minutes (%)	Percentage of UA patients, where an ECG is done within 10 minutes after arrival at the hospital or in the Ambulance to the hospital	Number of patients (#1) with intervention condition ECG duration <=10 minutes divided by Number of patients (#1)
Process	GRACE-score documented in EPR (%)	Percentage UA patients where the GRACE-score is documented in the EPR	Number of patients (#1) with data element condition "GRACE_documented" = true divided by Number of patients (#1)
Process	Diagnosis and Risk assessment on basis of clinical history, physical examination, ECG and biomarkers? (%)		Number of patients (#1) with data element condition "diagnosis_based_on_history_etc" = true divided by Number of patients (#1)
Process	Treatment decision on basis of risk assessment (i.e. GRACE-score)		Number of patients (#1) with data element condition "treatment_based_on_risk_assessment" = true divided by Number of patients (#1)
Process	CAG scheduled within time frame of treatment decision		Not enough information to formulate indicator.
Process	Door-to-needle time	Time between arrival of the patient at the hospital and the moment the PCI is conducted. Only for patients that are treated with emergency!	Average time difference between two interventions (#5), i.e. arrival and PCI with data element condition "emergency" = true
Process	Discharge from 7 West	Percentage of patient that is discharge from 7 West and hasn't had a CABG	Number of patients (#1) with data element condition "discharge from 7 west" = true AND "CABG" = false divided by Number of patients (#1)
		Percentage of patient that is discharge from CCU and hasn't had a CABG	Number of patients (#1) with data element condition "discharge from CCU" = true AND "CABG" = false divided by Number of patients (#1)

Domain	Performance indicator	Description	Formulation
Process	Throughput times		Average Throughput time of pathway (#6)
Process	Cardiologist seen on day 1 (%)	Percentage of patients that see a cardiologist on the First day of their hospitalization	It is assumed that the patient seeing a cardiologist is documented as data element called "see_cardiologist_on_first_day". Thus the indicator is formulated as Number of patients (#1) with data element condition "see_cardiologist_on_first_day" = true divided by Number of patients (#1)
Financial	DOT properly checked		Number of patients (#1) with data element condition "proper_DOT_check" = true divided by Number of patients (#1)

APPENDIX G Performance Measurement System Features

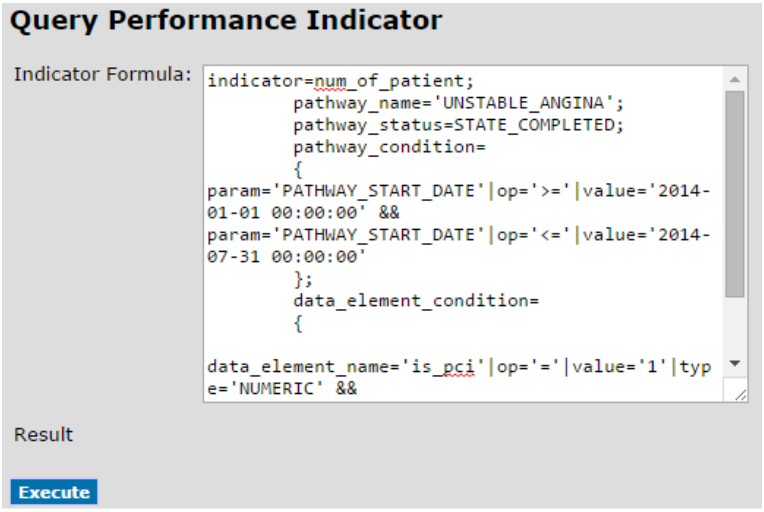


Figure 42 Feature query performance indicator

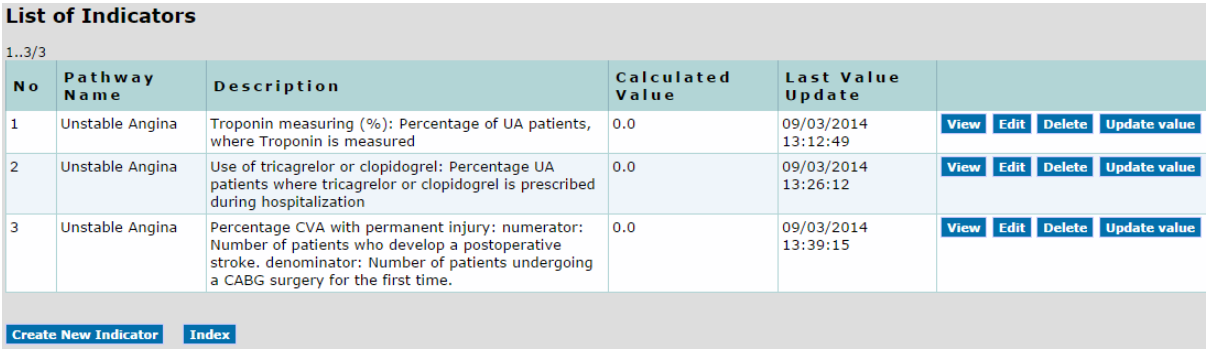


Figure 43 Feature create, view, edit and delete indicator

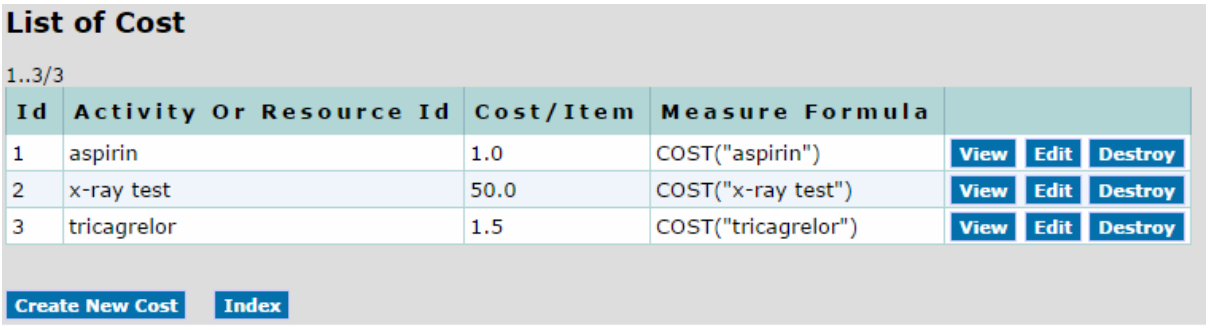


Figure 44 Feature Create, view, edit and delete cost

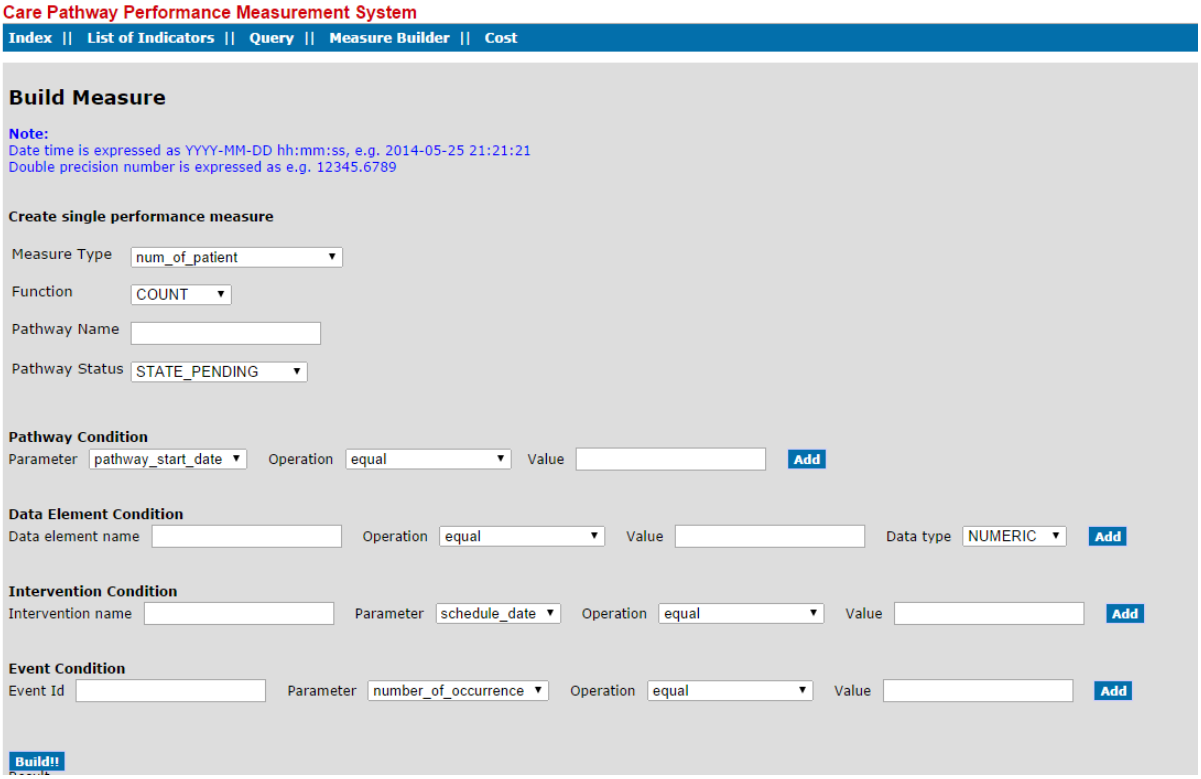


Figure 45 Feature build measure

APPENDIX H Weaning Protocol

The flowchart of weaning protocol is shown in Figure 46.

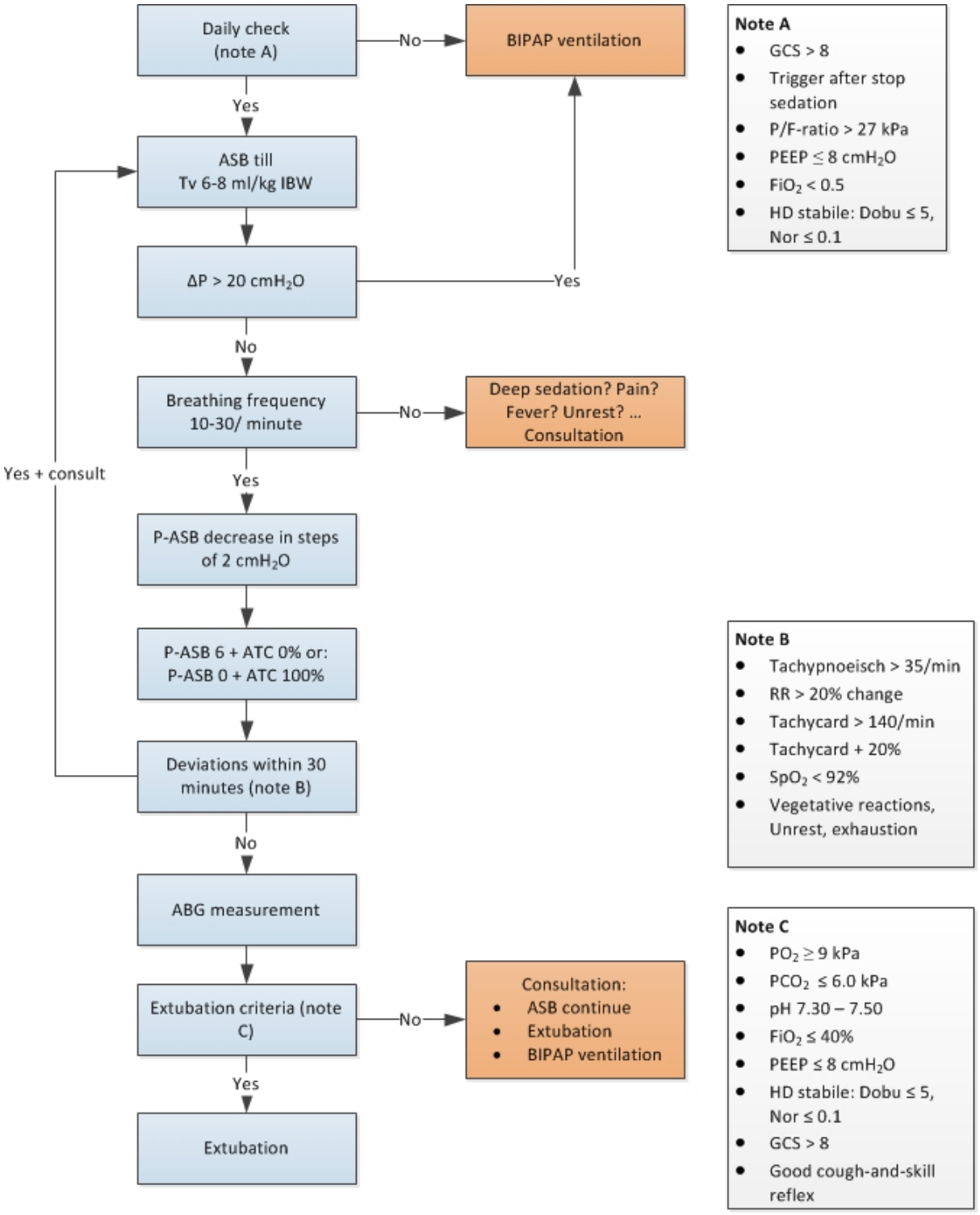


Figure 46 Original weaning protocol (Boere, 2013)

APPENDIX I Test Data

Table 20 Data elements for test

No.	GCS_A	PEEP_A	FIO2_A	TACHYPNOEISCH_B	TACHYCARD_B	SPO2_B	PO2_C	PCO2_C	PH_C	FIO2_C	PEEP_C	GCS_C	deviation_num
1.	14	10	81	38	186	83	11	4	7.4	97	6	9	3
2.	5	5	67	30	176	85	8	5	8	73	9	3	2
3.	9	6	61	13	198	83	11	4	7.4	43	4	9	4
4.	10	7	100	29	193	81	9	4	7.6	24	5	3	5
5.	7	7	40	37	144	84	11	7	7.2	59	7	15	1
6.	7	9	56	13	162	81	7	6	7.4	20	6	11	0
7.	6	7	24	13	193	95	9	6	7.4	39	6	12	3
8.	13	5	50	16	191	84	9	4	7.4	96	6	11	1
9.	11	8	23	31	170	98	10	4	7.2	92	9	8	4
10.	6	10	93	34	124	97	10	5	8	50	7	10	3
11.	14	4	32	22	199	81	9	7	7.1	63	4	15	5
12.	14	6	88	14	127	99	10	5	7.6	21	8	10	0
13.	13	10	35	36	177	89	8	7	7.6	27	9	7	0
14.	11	7	25	28	130	92	10	5	7.3	25	7	12	5
15.	11	5	51	34	179	97	10	4	7.5	33	6	10	5
16.	10	7	54	21	189	100	11	4	7.4	57	7	12	3
17.	15	6	77	26	135	99	8	5	7.1	54	7	3	4
18.	15	6	35	38	152	87	10	4	7.6	79	6	13	2
19.	7	5	46	34	158	93	11	6	7.3	39	4	7	0
20.	6	5	85	20	151	95	9	5	7.4	39	7	9	4
21.	9	5	74	23	137	96	11	5	7.4	35	6	10	2
22.	3	8	56	11	157	96	10	7	7.9	50	5	15	0
23.	3	5	26	36	117	99	10	4	7.9	74	4	8	5
24.	8	8	43	37	196	84	11	7	7.5	20	8	11	4
25.	5	10	35	31	179	100	8	4	7.2	27	10	10	2
26.	15	8	20	26	127	84	12	4	7.5	45	7	10	1
27.	3	5	51	18	177	97	10	4	7.2	60	6	10	4
28.	15	8	56	15	176	95	11	4	7.4	44	7	9	3
29.	3	9	28	28	179	84	7	6	8	83	8	15	5
30.	15	7	62	37	163	93	11	7	7.8	41	7	14	3
31.	7	9	84	22	129	87	8	4	7.6	97	9	3	1
32.	15	7	86	39	128	88	7	6	7.1	34	4	9	4
33.	3	5	94	35	193	81	7	6	7.2	22	6	6	4
34.	5	10	59	18	129	80	9	7	7.4	27	6	9	5
35.	3	6	37	32	123	93	11	6	7.5	28	8	4	5
36.	8	8	68	13	200	83	9	4	7.4	28	7	12	1
37.	10	4	81	33	173	99	11	5	7.2	49	8	15	4
38.	14	5	95	33	133	92	7	4	7	64	8	11	0

39.	8	6	49	17	191	87	8	6	7.6	76	4	3	3
40.	10	6	63	20	111	95	10	5	7.6	82	8	13	5
41.	7	9	34	28	187	90	10	5	7.5	36	5	8	4
42.	6	5	34	27	115	93	8	7	7.3	69	4	4	0
43.	6	4	90	14	168	96	11	5	7.5	50	7	11	0
44.	8	7	32	16	133	80	11	7	7.8	100	5	10	4
45.	6	7	70	12	140	99	11	4	7.9	54	7	6	2
46.	7	4	77	31	157	85	11	5	7.3	26	6	10	4
47.	8	8	23	33	108	98	8	4	7.7	80	8	7	4
48.	13	6	83	24	114	100	10	7	7.8	39	10	3	2
49.	14	6	80	10	127	84	8	5	7	93	9	3	2
50.	10	10	73	10	151	99	7	7	7.1	85	9	7	3

APPENDIX J Weaning Process KPI Expression

1. **Throughput time**

```
<FUNCTION>("measure='pathway_throughput_time';
pathway_name='weaningprotocol.weaning';
pathway_status='STATE_COMPLETED';")
```

Observed values: Average, Min, Max

2. **BIPAP time**

```
<FUNCTION>("measure='intervention_execution_time';
pathway_name='weaningprotocol.weaning';
pathway_status='STATE_COMPLETED'; intervention_name='BIPAP
ventilation';")
```

Observed values: Average, Min, Max

3. **ASB time**

```
<FUNCTION>("measure='intervention_execution_time';
pathway_name='weaningprotocol.weaning';
pathway_status='STATE_COMPLETED'; intervention_name='ASB
ventilation';")
```

Observed values: Average, Min, Max

4. **Time to reach 36° C**

Cannot formulate indicator.

5. **BIPAP threshold values fulfillment (NOTE A)**

```
COUNT("measure='num_of_patient';
pathway_name='weaningprotocol.weaning';
pathway_status='STATE_COMPLETED';
data_element_condition={
data_element_name='GCS_A'|op='>'|value='8'| type='NUMERIC' &&
data_element_name='PEEP_A'|op='<='|value='8'| type='NUMERIC' &&
data_element_name='FIO2_A'|op='<'|value='50'| type='NUMERIC'
}")
/
COUNT("measure='num_of_patient';
pathway_name='weaningprotocol.weaning';
pathway_status='STATE_COMPLETED';
")
```

6. **Number of steps to decrease ASB**

```
<FUNCTION>(" measure='num_of_intervention';
pathway_name='weaningprotocol.weaning';
```

```
pathway_status='STATE_COMPLETED'; intervention_name='Decrease P-
ASB'>; ")
```

Observed values: Average, Min, Max

7. **Extubation threshold values fulfillment (NOTE C)**

```
COUNT("measure='num_of_patient';
pathway_name='weaningprotocol.weaning';
pathway_status='STATE_COMPLETED';
data_element_condition={
data_element_name='PO2_C'|op='>='|value='9'|type='NUMERIC' &&
data_element_name='PCO2_C'|op='<='|value='6.0'|type='NUMERIC' &&
data_element_name='PH_C'|op='>='|value='7.30'|type='NUMERIC' &&
data_element_name='PH_C'|op='<='|value='7.50'|type='NUMERIC' &&
data_element_name='FIO2_C'|op='<='|value='40'|type='NUMERIC' &&
data_element_name='PEEP_C'|op='<='|value='8'| type='NUMERIC' &&
data_element_name='GCS_C'|op='>'|value='8'| type='NUMERIC'
}")
/
COUNT("measure='num_of_patient';
pathway_name='weaningprotocol.weaning';
pathway_status='STATE_COMPLETED';
")
```

8. **Number of observed deviations within 30 minutes**

```
<FUNCTION>("measure='data_element_value';
pathway_name='weaningprotocol.weaning';
pathway_status='STATE_COMPLETED';
data_element_name='deviation_num';data_type='NUMERIC';value_type=
'LAST'")
```

Observed values: Average, Min, Max