



Combi-BP: Automating the Data-Oriented Optimization in Business Processes. From declarative to executable models.

María Luisa Parody Núñez, 28.817.015-Q
lparody@us.es

Supervised by
Prof. Dr. María Teresa Gómez López
Prof. Dr. Rafael Martínez Gasca



Departamento de
Lenguajes y Sistemas Informáticos
Universidad de Sevilla

Thesis Dissertation submitted to the Department of Computer Languages
and Systems of the University of Seville in partial fulfilment
of the requirements for the degree of Ph.D. in Computer Science.
(Thesis Dissertation)

Abstract

One of the main objectives of a business expert is to model the business goals of an enterprise process. Several languages have been created to describe the necessary activities to achieve the objective, especially in the business process context. These languages can be divided into imperative and declarative ones. Declarative languages tend to be used when the specific model is unknown, being possible to describe what has to be done instead of how. Otherwise, imperative languages permit to describe how the things have to be done and then, the imperative models can be executed in any Business Process Management System (BPMS). The declarative descriptions are more flexible, since they permit to describe the model in a more relaxed way, which means that various process executions can follow the same declarative description. However, both paradigms are focused on the activities order description, but unfortunately, the data perspective is missed. Furthermore, the optimization of a business goal which depends on the exchanged data during the execution of the business process has not been included in previous proposals. There are no solutions that allow the business experts to describe nor execute a declarative description where the executed model depends on the exchanged data between the involved activities in each instance.

In this thesis dissertation, an approach to support this data-oriented optimization in business process is presented. A data-oriented optimization problem is a process whose main purpose is to obtain the best business product. In order to obtain this business product, the process must combine several activities by taking into account the existing data-structure and data-value dependencies. Both kind of dependencies are established by a set of constraints that relate the data (consumed and provided by the activities) and the data given by the customer. Therefore, the BPs under the scope of our research are those which are centred on developing sound data in business processes, analysing how data-structure and data-value dependencies can affect the correct business process execution. However, if the data provided at runtime for the activities that conform the model have not got enough level of quality, then business process will not be successfully executed.

The base of the proposal is focused on the combination of the advantages of both paradigms: the flexibility of the declaratives, and the automatic execution in a BPMS of the imperatives. On the one hand, we want to describe a flexible model using a declarative description where the exchanged data and an optimization objective are included. In the other hand, this declarative model must be executed in a generic business process management system with the aim of support any instance of the process. Therefore, how the declarative description can be transformed into an imperative business process is

developed. The transformation methodology that we propose is based on Model-Driven Architecture. Firstly, the declarative is transformed into an imperative which takes into account the data-structure dependencies. The flexibility of the declarative specification is kept thanks to the use of Constraint Programming. On the other hand, the resulting imperative model is enriched with new intelligent techniques, also based on Constraint Programming, in order to solve the data-value dependencies. Finally, a methodology and an implementation are developed in order to make the business process aware of the data-quality aspects.

Contents

Contents	iii
List of Figures	vii
List of Tables	ix
I Preface	1
1 Introduction	3
1.1 Context and Motivation	3
1.2 Problem Statements	6
1.3 Contributions	8
1.4 Thesis Context and Published Results	11
1.5 Roadmap: Structure of the Thesis	14
II Foundations	17
2 Foundations	19
2.1 Business Process Management	19
2.1.1 Concepts	19
2.1.2 Business Process Management Life-cycle	20
2.1.3 Business Process Perspectives	21
2.1.4 Process-Aware Information System	22
2.2 BP Imperative Description	23
2.2.1 Analysis of Imperative Languages	24
2.2.2 Business Process Model and Notation (BPMN)	27
2.3 BP Declarative Description	29
2.3.1 Characteristics	29
2.3.2 Analysis of Declarative languages	32
2.3.3 Comparative	34
2.4 Constraint Programming Paradigm	35
2.4.1 Constraint Programming Concepts	35
2.4.1.1 CSP Consistency	36

2.4.1.2	CSP Search Algorithms	36
2.4.2	Constraint Optimization Problems	37
2.4.2.1	COP Search Algorithms	38
2.4.3	Distributed Constraint Satisfaction Problem	38
2.4.3.1	Algorithms for solving DisCSP	39
2.5	Data Quality Management in Business Process	40
2.5.1	Data Quality Management Concepts	40
2.5.2	High-level and Low-Level DQM Activities	42
2.5.3	Data Quality Dimensions	43
2.6	Case Study: Trip Planner	44
III Contribution I: Combi-BP Specification		47
3	Specifying Data-Oriented Optimization in Business Processes	49
3.1	Context and Motivation	49
3.2	Formalization of Data-Oriented Optimization in BPs	50
3.2.1	Formalization Applied to the Trip Planner	53
3.3	Data-Oriented Optimization Declarative Language (DOODLE)	55
3.3.1	DOODLE Applied to the Trip Planner	55
3.4	BPMN Extension for Data-Oriented Optimization processes	57
3.4.1	Metamodel for the declarative sub-process	59
3.4.2	CombA Sub-Process Definition as Declarative Component	60
3.4.3	CombA Sub-Process Operational Semantics	62
3.4.4	CombA Sub-Process Handling Events	62
3.4.5	CombA Sub-Process Execution Semantics	63
3.4.6	BPMN Editor including CombA Sub-Process	64
3.4.7	BPMN Extension Applied to the Trip Planner	65
3.5	Related work	67
3.6	Summary and Discussion	68
IV Contribution II: Combi-BP Transformation		71
4	Configuration of an Imperative Business Process to minimize the execution time according to Data Dependencies	73
4.1	Context and Motivation	73
4.2	Configuration System Description	76
4.2.1	Relation between Data Dependencies and Imperative Models	77
4.3	Automatic Configuration from Declarative to Imperative Model	78
4.3.1	Creating a COP from the declarative model	79
4.3.2	Transformation of the COP results into a BP Imperative Model	81
4.4	Configuration applied to the Trip Planner	87
4.5	Related work	88
4.6	Summary and Discussion	89

5	Creating an Imperative Model to Optimize the Business Process Outcome	91
5.1	Context and Motivation	91
5.2	The White-Box Approach	93
5.2.1	How to transform a White-Box Specification into an Executable Business Process	93
5.2.2	White-Box Model Transformation applied to the Trip Planner . . .	94
5.2.3	Empirical Evaluation	96
5.2.3.1	Experimental Design	96
5.2.3.2	Experimental Result	97
5.3	The Black-Box Approach	97
5.3.1	Coordinator Algorithm	99
5.3.2	Experimental Results	100
5.4	Related work	102
5.5	Summary and Discussion	103
V	Contributions III: PAIS-DQ	105
6	PAIS-DQ	107
6.1	Context and Motivation	107
6.2	Detailing a Case Study	108
6.3	PAIS-DQ Description	110
6.3.1	PAIS-DQ Architecture	110
6.3.2	Data Quality Management Activities	112
6.3.2.1	How High-Level DQ Management can affect the BP Model	112
6.3.2.2	How Low-Level DQ Management Activities can affect the BP Model	112
6.3.3	Data Quality Layer Functionalities	113
6.4	PAIS-DQ-HOW: Methodology to use PAIS-DQ	114
6.4.1	Business Process Design	115
6.4.2	Data Quality Layer and System Configuration	115
6.4.2.1	Configuration of High-Level DQ Management Activities .	116
6.4.2.2	Configuration of Low-Level DQ Management Activities . .	116
6.4.2.3	Changes to the BP to make it DQ aware	117
6.4.3	Business Process Execution	117
6.5	PAIS-DQ Applied to the Case Study	117
6.5.1	BP Design: Flight Search Process	119
6.5.2	DQ Layer and System Configuration	119
6.5.2.1	Configuration to Control Data Quality Levels	119
6.5.2.2	Data Quality Items	119
6.5.2.3	Low-Level DQ Activities to Control DQ Configuration . .	120
6.5.2.4	Low-Level DQ Activities to DQ Control Implementation .	121
6.5.2.5	<i>Flight Search</i> Process Changes	122
6.5.3	BP Execution	123

6.6	Related work	123
6.7	Summary and Discussion	124
VI	Conclusions and Future Work	125
7	Final Remarks	127
8	Directions of Future Work	131
VII	Appendices	135
A	Abbreviations	137

List of Figures

- 1.1 Investment Decision Process Example. 5
- 1.2 Combi-BP Framework. 9
- 1.3 The Structure of the Thesis. 15

- 2.1 BPM Life-cycle. 21
- 2.2 PAIS Framework Architecture. 23
- 2.3 Graphical solution of a CSP system 36
- 2.4 Trip Planner Example (OMG, 2011a). 44

- 3.1 External and Internal Components of the Data-Oriented Optimization Process 51
- 3.2 Example of the trip planner described using DOODLE 58
- 3.3 Data-oriented optimization process included in the Trip Planner Process. . 58
- 3.4 Extension of BPMN 2.0 Sub-Process Metamodel. 59
- 3.5 Collapsed and Expanded representation of CombA Sub-Process. 61
- 3.6 A collapsed and expanded CombA Sub-Process with Timer Event. 63
- 3.7 Vocabulary in CombiS-BP Editor. 65
- 3.8 CombiS-BP Editor. 66
- 3.9 Expanded Search Travel CombA Sub-Process. 66
- 3.10 DOODLE Graphical Components. 69

- 4.1 Imperative BP Representation for Data-Oriented Optimization BP. 74
- 4.2 Imperative BP Representation for Data-Oriented Optimization BP. 75
- 4.3 Imperative BP Representation for Data-Oriented Optimization BP. 75
- 4.4 Configuration System Architecture. 76
- 4.5 “Flight Search” and “Car Rental 1 Search” Model Possibilities 77
- 4.6 Configuration from Declarative BP Model to Imperative BP Model 78
- 4.7 Flexibility of the BP in terms of the execution time of the Activities 80
- 4.8 Trace example of the Algorithm 2 to create a *BPMN – Graph* using the COP results 83
- 4.9 Parallel Relationship 84
- 4.10 Exclusive Relationship 84
- 4.11 Inclusive Relationship 86
- 4.12 From graph to BPMN Model Example 86
- 4.13 Trip Planner Model Result 88

- 5.1 White box transformation. 93

5.2	Choco Script Task Configuration in Bonita Open Solution TM	95
5.3	Number of possible combinations for each test case.	96
5.4	Memory used for the tests of the example.	97
5.5	Time used for the tests of the example.	98
5.6	Black box approach schema.	98
5.7	Backtracking Tree Structure.	100
5.8	Branch and Bound Tree Structure.	101
5.9	Number of calls to <i>Calculate_Objective()</i> by Algorithm 1 (<i>Backtracking</i>) and by Algorithm 2 (<i>Branch-and-Bound</i>).	102
6.1	Illustrative Example: Flight Search process.	109
6.2	Framework for Data Quality Management.	111
6.3	Data Quality Control by including new decision rules in a gateway.	113
6.4	Data Quality Assurance by inclusion of new branches.	114
6.5	Data Quality Action Performance in an external service.	115
6.6	Data Quality Layer services.	115
6.7	Add a <i>Data Quality Connector</i> to an activity with Bonita Open Solution TM	120
6.8	<i>Data Quality Connector</i> configuration for the activity <i>Request Flight From Provider 1</i>	121
6.9	Fragment corresponding to the completeness dimension given by I8K Ar- chitecture.	121
6.10	Connectors in Bonita Open Solution TM for the activity <i>Request Flight From Provider 1</i>	122
8.1	Smart cities requirements.	133

List of Tables

- 2.1 Main BPMN Elements used in this thesis dissertation 28
- 2.2 Declarative Languages Comparative 35
- 2.3 Actions depending on DQ Requirements 43

- 3.1 Internal Components associated to the activities of the declarative model . 56
- 3.2 External Components of the declarative model 57
- 3.3 CombA Sub-Process model attributes 61
- 3.4 CombA Task model attributes 62
- 3.5 Declarative Languages Comparative including DOODLE 68

- 4.1 Declarative model and COP elements relationship 80
- 4.2 Type of gateway decision 86

- 5.1 Constraint Optimization Problem for outcome data optimization 95

- 6.1 DQ Layer in PAIS-DQ 111
- 6.2 PAIS-DQ-HOW Methodology 118
- 6.3 Data Quality Requirements for *Request Flight from Provider 1* activity
output 119

- A.1 Abbreviations 138

Part I
Preface

Chapter 1

Introduction

*It is only possible to move forward when you look away.
We can only make progress thinking big.*

José Ortega y Gasset

1.1 Context and Motivation

A business process consists of a set of activities that are performed in coordination within an organizational and technical environment. These activities jointly perform a business goal (Weske, 2007). In business process management theory of the last years, a process-oriented perspective has been considered the shell on organizational (re)structuring. Nowadays, organizations still experience difficulties in applying this process-oriented perspective to the design and maintenance of their information systems. Many of the problems deal with the imperative representation of business processes, since they contain unsuitable information for computer systems to provide flexible and automated business process support.

Typically, business processes are specified in an imperative manner, (e.g., by indicating that activities A and B are executed sequentially or in parallel). This imperative specification allows business experts to describe relationships between activities and to transform the process into an executable model supported by a commercial Business Process Management System (BPMS). Imperative descriptions define exactly how things have to be performed. But sometimes, a business process may be exposed to different environments and subjected to many conditions in which not always a sequence of activities can be described at design time. This is the reason why several authors have proposed languages to define business process as declarative models (Pesic and van der Aalst, 2006; Rychkova et al., 2008a,b; Sadiq et al., 2005). A declarative representation takes into account the business concerns that govern the business process, describing the policies of the organization. These declarative languages tend to be used to describe the possible execution order of the activities, allowed or prohibited, instead of the exact order of the activities. Related to the capacity to execute a declarative business process, although

there are available solutions to execute declarative descriptions, they only provide guidelines or recommendations on how a concrete instance of a declarative language must be executed, not existing commercial BPMSs that support the declarative models.

Accordingly, the main duties of a business expert are the design and modelling phases of the Business Process Management life-cycle (Weske, 2007). Business experts have to define a model which represents the business goals of an enterprise process, to be carried out during the execution phase. Depending on the type of problem, either imperative or declarative description is more appropriate. When the order of activities is clear and known, an imperative model is more appropriate, since it is more understanding and can be automatically deployed and executed using a BPMS. But when the order of activities is not exactly known at design time, or it is difficult to determine, a declarative model is better, since it enables a more flexible and adaptable solution. Therefore, the ideal scenario is the combination of declarative and imperative descriptions in order to attain the best of both worlds. This first entails the flexibility of the declarative definition, and secondly, the easy transformation of a model into an executable process deployed in a BPMS with workflow support.

However, not only is the order of the activities crucial in the successful execution of a business process, but the data exchanged between the activities to and the data dependencies between the activities could be significant too. The existing proposals, both declarative and imperative, are specially focused on the order of the activities, not paying enough attention to process data. For these proposals, data can be used to establish the order of execution through its association with conditions in the form of decision data or business rules, whose evaluation could determine whether an activity should be executed or not. Nevertheless, they are neither concerned with the data exchanged during the process instantiation, and how the data dependencies can affect the activity order. It would establish and limit the execution order of the activities, and thus would lead to the successful and optimal execution of the process.

Generally, there are different types of goals in the business processes: (i) to obtain the best product to satisfy the customer requirements, (ii) to reduce the time and/or cost of the process execution, or (iii) to simply perform a task, such as the management of documentation, or provide products. The latest group does not need any optimization, just the modelled of the process. Typically, the declarative and imperative models have been used to obtain the best order of the activities to reduce the time or the execution cost of each instance. However, a great problem is presented when the aim of the process focuses on a combination of activities in order to obtain the best business product (outcome data), by means of the best combination of data belonged to the data-flow.

An example where the exchanged data during the instance is important to optimize the business product, is when a customer has to decide how much and where invest a quantity of money (see Figure 1.1). If the customer wants to invest an amount of money in funds for a certain period of time, and can choose between different entities and/or assets in order to maximize the profit, (s)he needs to search by hand the specific amount of money to invest in each entity and/or asset. The entities and/or assets are represented by activities in the business process. Each activity offers a diversified selection of funds with different risks and profits, and depending on the specific amount of money, these characteristics

varies. This variation involves a continuous consult by the customers to the entities, in order to know the profit returned given the different amounts of money. In addition, it is necessary to consider, that the amount of money that is invested in an entity, will influence the amount that is spent in another entity, and thus, the total profits obtained. In other words, this example illustrates the importance of the decision related to which combination of input data (amount of money) would return the best business product (highest profit). In this case, the model could be known, being possible to execute the activities in a parallel manner, but the specific input data for each activity are unknown. The problem here is to decide in design time, how to model a business process to find the best combination of values for the data input of the activities, takes into account the data dependencies, and also achieves the customer requirements at runtime. It is important to highlight that, although the BP requirements are known and can be easily specified, neither imperative nor declarative representations can be used to describe this problem.

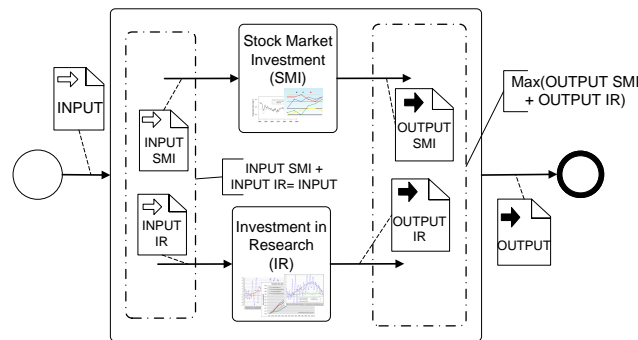


Figure 1.1 – Investment Decision Process Example.

We focus on these type of data-oriented optimization processes, whose main purpose is to find a set of data that need the activities of the process to optimize an objective function. On the one hand, there is a data-structure dependency, since the constraints establish a possible order between the activities by means of their data relationships. This order is not easily to establish at design time without any help. On the other hand, there is also a data-value dependency, which establish the data requirements necessary to obtain the optimal outcome in each instance. Therefore, the aim of this thesis dissertation is to help business experts to design and implement a data-oriented optimization process. The business experts know the description of the problem in a declarative way, but they do not know how to transform it into an imperative model. The complete proposal shields the business experts to unnecessary implementation details, and helps them to describe the model in a more flexible and adaptable way. The modelling of the business process and the deployment in a Business Process Management System is going to be faster and will save personal and time in companies, since it helps to face problems of optimization in an easy, automatic and flexible way.

As was commented, the main part of the scope of our research is the data that generate and compose the final product in business processes. We consider them critical, and

essential for the business process which are centred on developing sound data products rather than the sound execution of the processes. Furthermore, the management of data with the adequate level of quality also constitutes a key value for the successful execution of these processes. In order to make organizations aware of the importance of data quality, a data quality management plan should be implemented that covers the main data and data-quality requirements in the business process. Following with the investment example, a set of quality requirements can be demanded with the aim of making a right decision. For example, a percentage of the profits from an investment are taxes for the Government, we could establish that as soon as this information is given by the entities (whether the profits include these taxes or not), the information will be complete. In this thesis dissertation, we propose some mechanisms to enrich the business processes in order to control and/or assure the data quality requirements, such as the completeness used in the example.

In the next sections, we detail the problem statements, and how they are addressed in this thesis dissertation.

1.2 Problem Statements

This thesis seeks the enhancement of data-oriented optimization problems in business processes by proving automatic transformation from a declarative model, focused on data, into a flexible imperative model. This challenge can be addressed by considering how the data-oriented model can be described in a declarative way; how an imperative model can be equivalent to a declarative description; and how data quality requirements can affect the business process model.

How the data-oriented model is described in a declarative way?

Business process modelling constitutes an essential and crucial task in the Business Process Management life-cycle. Business process models gather a large amount of information and structures (van der Aalst, 2012) (e.g. activities, control-flow, data objects, and time constraints), since it greatly influences the remaining phases of the cycle.

There are a significant number of researches that detect the necessity to include the data description into the business process model. Although certain imperative languages have included description capacities in an imperative way for the exchanged data (OMG, 2011b), new extensions have still been proposing, such as in (Gómez-López and Martínez Gasca, 2010a; María Teresa Gómez-López and Gasca, 2014; Meyer et al., 2011, 2013). This concern in the imperative languages remains much less relevant in declarative scenarios, which are more centred on the order of activities. The role of data in declarative languages has not been very relevant, mostly limited to describe the execution or not of an activity, depending on the value of a variable of the data-flow.

There is a significant number of declarative languages to describe business processes. They tend to be used when business processes need to be flexible and adaptable, being not possible to use an imperative description. The declarative languages define what has to be done instead of how, since the specific order between the activities cannot be known at design time in an imperative manner. A declarative representation takes into

account the business concerns that govern the business process, described by means of declarative policies of the organization, specifying the possible execution order of the activities, allowed or prohibited.

After the analysis of the literature, we have concluded that how the data can affect the business processes has not been resolved. While the declarative languages cover the flexible description of the activity order, and the imperative languages study how to describe the data, neither of them deal with how data can be included in declarative model. Depending on the data instantiated, the creation of one model or another could be better, since, for example, the model will influence the selection of the best data input of the activities to satisfy the process requirements, as the example of the investment presented above. In that case, although the business process requirements are known at design time, the imperative representation is extremely difficult to determine since it must be flexible enough to support any instance. Unfortunately, none of the existing declarative and imperative languages are worried about how to represent neither to support the exchange of data with the aim of optimize the business product, which influences the business process model.

How an imperative model can support a declarative description?

Some of the benefits of the declarative models are the flexibility and the ease of writing (Fahland et al., 2009). However, the most important disadvantage is that, it is not possible to deployed it in a commercial BPMS. On the other hand, the imperative models are more rigid, although closer to executable models.

A significant number of tools have been developed to transform imperative models into executable model in an easy way. As soon as a model can be executed, more usable and applicable is for organizations. In order to support the deployment and execution of a business process model, a Business Process Management System is necessary (Weske, 2007). There are several Business Process Management System on the market that enable the analysis, definition, execution, monitoring, and control of processes. Generally, the four critical components of a Business Process Management System are: process engine, business analysis, content management, and collaboration tools. Furthermore, these four components are also focused on building a bridge between business experts and IT since traditional Business Process Management engines are focused on non-technical people only. Thus, these Business Process Management Systems offer process management features in a way that both business and IT users could use.

While the imperative languages are easily transformed into executable, it does not happen the same with the declarative languages. Although some tools are available to execute declarative descriptions, none provide anything more than guidelines or recommendations about which activity should be executed at specific moments of an instance, since it remains impossible to execute or deploy them in a Business Process Management System environment. Which means that the declarative descriptions, and the systems that support them, are not enough to be used in the daily work of a company.

Focused on data-oriented optimization processes, an executable model will be desirable as well, not only to determine which activity to execute at any moment, but also to

integrate the business process data requirements in the executable model, by means of solving the data-structure dependencies. The executable model needs to ascertain the specific values for each instance that optimize the outcome data of the process by means of solving the data-value dependencies.

Since declarative and executable models are conceptually too separated, we propose the creation of an intermediate imperative model combined to Constraint Programming. Thanks to the use of Constraint Programming paradigm, data-structure and data-value dependencies can be taken into account, while maintaining the flexibility of the declarative models.

How data quality requirements are included in business process?

The data that generate and compose the final product are considered critical, and is essential for the business process (Wang, 1998). Among other factors, it can be said that the success of an instance of the processes is grounded in the quality of the data used. In order to gain advantages from data quality, organizations need to introduce mechanisms aimed at checking whether data satisfy certain data-quality requirements at runtime. Although certain data-quality related studies could be used at the design phase, such as (Cappiello et al., 2013; Pipino et al., 2002; Rodríguez et al., 2012), there is a lack of proposals for their *system configuration* and *process enactment* phases. The scope of data-quality requirements should already has been dealt with by business experts. On the other hand, IT people should be able to implement the corresponding mechanism to satisfy the stated data-quality requirements at runtime. However, most of the existing problems in applying a data-quality management program lies on the modification of the business process model itself with data-quality aspects, besmirching and complicating the understanding, functionality and operability of the model.

1.3 Contributions

In order to face the challenges of this thesis, a framework for the Combination of Data in Business Process (hereinafter Combi-BP) has been developed in order to fill the gap of data-oriented optimization problems in BP. Combi-BP aims to aid and support business and IT experts towards the design and implementation of data-oriented optimization problems in an automatic way. The base of data-oriented optimization problems is the specification of (i) the data handle by the process and by each activity; (ii) the constraints that relate these data; and (iii) the objective function to be optimized.

The process modelled in Figure 1.2 shows the relation between the contributions of this thesis dissertation, and the necessary steps to transform a data-oriented optimization process into an imperative model. The different steps are represented as tasks within the process, and the input and output of each contribution are represented as data objects.

Firstly, the business process requirements are specify in a declarative model. This declarative model is focused on data optimization and represents the set of data, activities, constraints that relate the activities, and the objective function to be optimized. The language to model a data-oriented declarative models is the base of the first contribution.

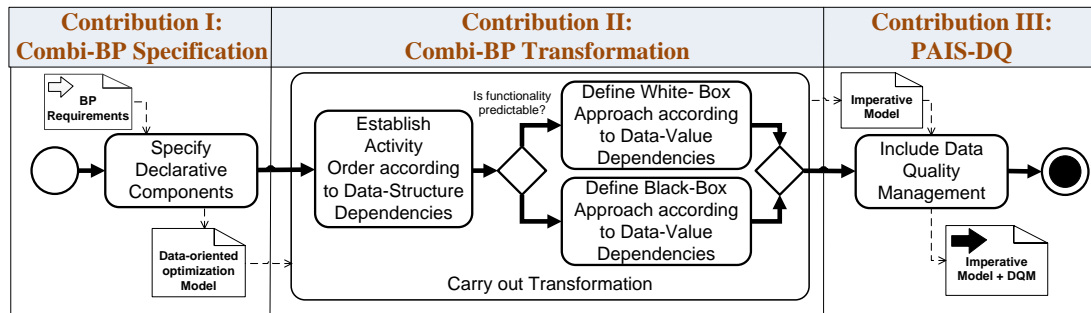


Figure 1.2 – Combi-BP Framework.

Secondly, how to carry out the transformation into a flexible imperative model is divided into various activities. In the first place, the establishment of an order between the activities by analysing the data-dependencies is necessary. The order must assure that the optimization is achieved and all the constraints are considered since these constraints establish the data-structure dependencies. However, in order to keep the flexibility of the declarative model, certain algorithms and data treatment must be included to solve the data-value dependencies. Whether the functionality of the activities could be predictable or not, two different transformations are possible. On the one hand, if the information provided in the business process requirements permits to ascertain the functionality of the activities without execute them, then a white box approach is applied. On the other hand, whether the information are insufficient to ascertain the functionality of the activities, and their execution are required, a black box approach is then applied. Even so, both models finish the transformation creating an imperative, flexible and closer to executable model. Finally, this imperative model is enriched with data-quality aspects in order to increment the trustworthiness of the data involved in and returned to the customer in the process.

The following subsections introduce each contribution with mode details.

Contribution I Combi-BP Specification: Data-Oriented Optimization Specification

As stated earlier, the existing declarative and imperative proposals fail to take into account how the exchanged data between the activities can affect the successful execution of the process. Our proposal provides a formalization of a business process data requirements to represent the outcome data optimization. The first contribution of this thesis dissertation focuses on the representation of this data-oriented optimization processes (cf. Chapter 3).

To this end, a Data-Oriented Optimization Language, called DOODLE, represents graphically a declarative model which includes the business process requirements referring to data description. A new point of view of declarative languages focused on data is presented, since priority is given to the significance of the information that flows through the process and between the activities to reach optimal outcome data in accordance to

customer requirements.

In order to combine the business process requirements, represented by this declarative language, with an imperative representation, we also propose an extension of BPMN 2.0 (OMG, 2011b) where a new type of sub-process and its associated marker is proposed. The aim of this new sub-process is to incorporate the declarative components of a data-oriented optimization process into an imperative model, and to find the best business product.

Contribution II Combi-BP Transformation: Business Processes Configuration and Outcome Optimization Enactment for Flexible Models

One of the main features of the optimization of data-oriented optimization processes is that the business process and the data requirements are known at design time. There are two main problems: (1) the business experts know what they want but not how to model it, since the imperative representation is extremely difficult due to the data dependencies and the objective function to be optimized; and (2) declarative models are good in the sense that they are flexible, but fail in their capacity to be executable, in opposite imperative models.

Consequently, the second contribution aims to provide the necessary support to configure and execute the declarative specification by means of transform it into a flexible imperative model (cf. Part IV).

Using the model of the first contribution, the data-structure dependencies are analysed, by means of the relationships between the activities. This analysis enables to find, in an optimal way, an equivalent imperative model, in accordance to the data dependencies between the activities. This equivalence, by means of imperative representation, is obtained thanks to the use of Constraint Programming paradigm and a set of developed algorithms. Since the declarative specifications can lead to several imperative models, we demand, to establish the best imperative model as the one that minimize the execution time of the instances. However, this imperative representation is not enough to maintain the declarative flexibility, since the data-value dependency and the optimal function to achieve the best outcome need to be obtained.

In order to solve this data-value dependencies, two different approaches are also developed in this second contribution. On the one hand, the called *White-Box* approach is applied whether the declarative model includes the necessary information to ascertain the output of the activities without execute them. This information enables to obtain the optimal values in a local manner by using constraint programming. On the other hand, a set of algorithms are developed to apply the called *Black-Box* approach. In that case, the activities must be executed several time in order to ascertain the optimal outcome data. The algorithms have to check every possible combination of data, and must guarantee that the optimal is achieved.

We have decided to use Constraint Programming since between its advantages are: It is a very mature area that has been applied to very different problems related to optimization, and with high level of complexity; it uses propagation techniques to reduce

in an efficient way the search space; there are a lot of tools and algorithms to model and solve problems; it permits a more easy definition of the data dependencies using a wide types of constraints, such as: implication constraints, disjunctive constraints, refied constraints, global constraints, and channeling constraints.

Contribution III: Data Quality Management in Business Process

One of the challenges in business processes is focused on identifying problems in order to improve the process. A well designed business process model cannot work correctly with incorrect data. A good way to avoid problems, related to the data managed, is by controlling and ensuring the quality of the data that flow through the process. To this end, the specification of data-quality requirements, and the implementation of certain activities focused on data-quality, must be introduced. Not only do we propose a theoretical solution, but we also define the steps to obtain an executable quality-aware business process (cf. Chapter 6).

These mechanisms, aimed at checking whether data satisfy certain data-quality requirements, attain the implementation of high-level data-quality management activities, such as control and assurance. In turn, these activities require the implementation of low-level data-quality management activities, such as measurement, assessment, and enhancement. This contribution proposes: (i) a methodology to guide developers in their task, which prevents organizations and customers from being exposed to unnecessary details on how the data-quality is managed; and (ii) a framework to suitably address the execution of specific high level data quality management activities.

1.4 Thesis Context and Published Results

This thesis dissertation has been developed in the context of the Quivir Research Group of the University of Seville under the scope of the following research projects:

- OPBUS: Improving business process quality by means of optimized and fault-tolerance technologies (P08-TIC-04095) funded by the Department of Innovation, Science and Enterprise of the Regional Government of Andalusia.
- TDiaCO-BPMS: Techniques for diagnosis, reliability and optimization in business process management systems (TIN2009-13714) funded by Spanish Ministry of Science and Technology.

The following papers have been published as either intermediate or directly results of the research findings presented in this thesis:

- Luisa Parody, María Teresa Gómez-López, and Rafael M. Gasca. **Data-Oriented Declarative Language for Optimizing Business Processes**. 22nd International Conference on Information System Development (ISD 2013). Sevilla (Spain), 2013. Full paper (In press). (CORE A)

- Ismael Caballero, Isabel Bermejo, Luisa Parody, María Teresa Gómez-López, Rafael M. Gasca, and Mario Piattini. **I8K: An Implementation of ISO 8000-1X0**. The 18th International Conference on Information Quality (ICIQ 2013). Little Rock (Arkansas, EEUU), 2013. Pp. 356 - 370. ISBN: 978-84-695-8310-4.
- Isabel Bermejo, Luisa Parody, Ismael Caballero, and Mario Piattini. **CONFIA Registration-2013. CONFIA COMpliaNt Framework ISO 8000 (eight thousAnd)**. Exp: CR-193-2013. This publication refers to the registration of our tool in the Intellectual Property Record of Ciudad Real (Spain).
- Luisa Parody, María Teresa Gómez-López, and Rafael M. Gasca. **Decision-Making Sub-Process to Obtain the Optimal Combination of Input Data in Business Processes**. IX Jornadas de Ciencia e Ingeniería de Servicios (JCIS 2013). Madrid (Spain), 2013. Pp. 17-31. ISBN: 978-84-695-8351-7.
- Isabel Bermejo, Luisa Parody, Ismael Caballero, María Teresa Gómez-López, and Rafael M. Gasca. **Gestión de Calidad de Datos en la Combinación de Actividades dentro del Marco de los Procesos de Negocio**. XVIII Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2013). Madrid (Spain), 2013. Pp. 195 - 208. ISBN: 978-84-695-8310-4.
- Luisa Parody, María Teresa Gómez-López, Rafael M. Gasca, and Angel Jesus Varela-Vaca. **CombiS-BP Editor Registration-2013**. This publication refers to the registration of our editor in the Intellectual Property Record of Andalusia (Spain).
- Luisa Parody, María Teresa Gómez-López, Rafael M. Gasca, and Angel Jesus Varela-Vaca. **CombiS-BP Editor: Combining Declarative and Imperative Languages in BP Modelling**. 7th IEEE International Conference on Research Challenges in Information Science (RCIS 2013). Paris (France), 2013. Pp. 663-664. ISBN 978-1-4673-2914-9. (CORE B)
- Luisa Parody, María Teresa Gómez-López and Rafael M. Gasca. **Extending BPMN 2.0 for Modelling the Combination of Activities That Involves Data Constraints**. 4th International Workshop on the Business Process Model and Notation (BPMN 2012). Viena (Austria), 2012. Pp. 68-82. ISBN 978-3-642-33154-1.
- Luisa Parody, María Teresa Gómez-López, Rafael M. Gasca, and Angel Jesus Varela-Vaca, **Improvement of Optimization Agreements in Business Processes Involving Web Services**, Communications of the IBIMA, Volume 2012 (2012), Article ID 959796, 15 pages DOI: 10.5171/2012.959796.
- María Teresa Gómez-López, Rafael M. Gasca, Luisa Parody, and Diana Borrego. **Constraint-Driven Approach to Support Input Data Decision-Making in Business Process Management Systems**. International Conference on Information System Development (ISD 2011). Edimbug (Scotland), 2011. Pp. 457 - 469. ISBN 978-1-4614-4950-8. DOI: 10.1007/978-1-4614-4951-5_37. (CORE A)

- Luisa Parody, María Teresa Gómez-López, Rafael M. Gasca, and Angel Jesus Varela-Vaca. **An Approach for Optimization Agreements in Business Processes based on Web Services**. 17th International Business Information Management Association Conference (IBIMA Conference). Milan (Italy), 2011. Pp. 183 - 194. ISBN 978-0-9821489-6-6. (CORE B)
- Luisa Parody, María Teresa Gómez-López, Rafael M. Gasca, and Diana Borrego. **Using Distributed CSPs to Model Business Process Agreement in Software Multiprocess**. 3rd International Conference on Agents and Artificial Intelligence (ICAART'11). Roma (Italy), 2011. Vol. 2, Pp. 434-438 ISBN: 978-989-8425-41-6. (CORE C)
- Luisa Parody, María Teresa Gómez-López, Rafael M. Gasca, and Diana Borrego. **Resolución de Acuerdos en Procesos de Negocio para Multiprocesos Software usando Programación con Restricciones Distribuidas**. Workshop de Apoyo a la Decisión en Ingeniería del Software (ADIS'10). Valencia (Spain), 2010. Vol. 4, No. 1, Pp. 53-64.

The following paper has been submitted to a relevant journal as direct results of the research findings presented in this thesis:

- Luisa Parody, María Teresa Gómez-López and Rafael M. Gasca. **Optimization of Outcome Data of Business Processes. From Declarative to Imperative Process Models**, submitted to Information and Software Technology Journal, Ed. Elsevier. (JCR-2013 1.522).
- Luisa Parody, María Teresa Gómez-López, Isabel Bermejo, Ismael Caballero, Rafael M. Gasca, and Mario Piattini. **PAIS-DQ: Extending Process-Aware Information Systems to support Data Quality in PAIS life-cycle**, submitted to European Journal of Information Systems, Palgrave Macmillan. (JCR-2013 1.558).
- Luisa Parody, María Teresa Gómez-López and Rafael M. Gasca. **Configuration of an Imperative Business Process according to Data Dependency Aspects**, submitted to Software & System Modeling Journal, Springer. (JCR-2013 1.250).
- Isabel Bermejo, Ismael Caballero, Luisa Parody, María Teresa Gómez-López, Mario Piattini, and Rafael M. Gasca. **Managing data quality in master data exchanged by means of ISO 8000-1x0**, submitted to IEEE Transactions on Knowledge and Data Engineering, IEEE Computer Society. (JCR-2013 1.676).

The author has also participated actively in other relevant contributions during the development of this thesis:

- Diana Borrego, Rafael M. Gasca, María Teresa Gómez-López, and Luisa Parody. **Contract-based Diagnosis for Business Process Instances using Business Compliance Rules**. 21st International Workshop in Principles of Diagnosis (DX'10). Portland, Oregon, USA, 2010. ISBN: 978-1-936263-02-8.

- Angel Jesus Varela-Vaca, Rafael M. Gasca, and Luisa Parody. **OPBUS: Automating Structural Fault Diagnosis for Graphical Models in the Design of Business Processes**. 21st International Workshop in Principles of Diagnosis (DX'10). Portland, Oregon, USA, 2010. ISBN: 978-1-936263-02-8.
- Diana Borrego, María Teresa Gómez-López, Rafael M. Gasca, and Luisa Parody. **Diagnosis de Errores en la Gestión de Procesos Software con Programación con Restricciones**. Workshop de Apoyo a la Decisión en Ingeniería del Software (ADIS'10). Valencia (Spain), 2010. Vol. 4, No. 1, Pp. 23-34.

The author has also made the following stays during the development of this thesis:

- Research stay in Ciudad Real, Spain, from 15 May 2013 until 15 August 2013 under Mario Piattini's supervision, and collaborating with Dr. Ismael Caballero. Mario Piattini is the head of Alarcos Research Group at the University of Castilla-La Mancha.
- Research stay in Trento, Italy, from 7 May 2012 until 5 August 2012 under Marco Pistore's supervision, and collaborating with Dr. Annapaola Marconi. Marco Pistore is the head of Service Oriented Applications Research Unit of the Fondazione Bruno Kessler (FBK).

In addition, the author has lead the following final degree projects related to this thesis:

- Final project entitled **l8K: Arquitectura de Servicios para la Gestión de la Calidad de los Datos: Una implementación de ISO 8000:2009-100**, realized by D. Isabel Bermejo Manzaneque, and also supervised by Dr. Ismael Caballero Muñoz-Reja. The project is related to the Engineering in Computer Science degree at the Computer Science School in Ciudad Real, University of Castilla-La Mancha, Spain. It was defended the 14th June 2013, achieving a score of honours (10).
- Final project entitled **Aplicación Web para la Especificación Declarativa de Procesos de Negocio**, realized by D. Juan Francisco Fernández Narváez. The project is related to the Technical Engineering in Computer Management degree at the Computer Science School in Seville, University of Seville, Spain. It was defended the 3th July 2013, achieving and score of outstanding B (8.5).

1.5 Roadmap: Structure of the Thesis

The structure of this thesis is illustrated as a business process diagram such as drawn in Figure 1.3. The thesis dissertation encompasses four main parts:

Part I: Preface. In this part, the research context, motivation, problem statements and research findings of the thesis are introduced. To conclude this part, a list of the most relevant contributions attained during the development of the thesis are presented.

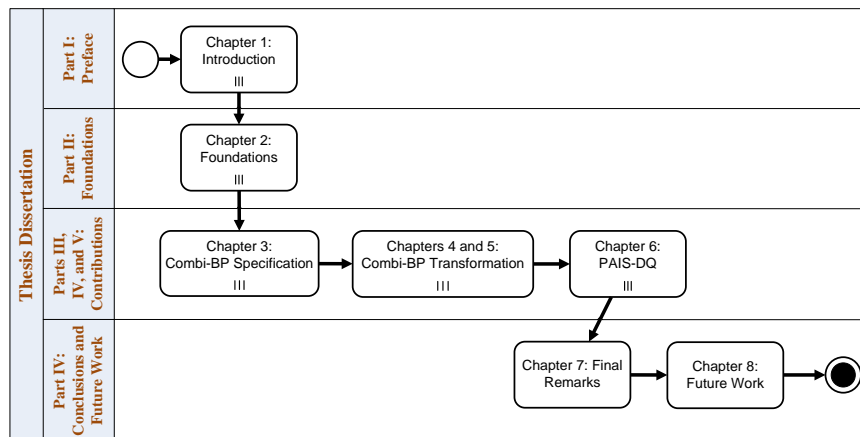


Figure 1.3 – The Structure of the Thesis.

Part II: Foundations. In this part, we introduce to the reader to the most relevant concepts in the different areas in which the thesis has been developed. Section 2.1 introduces the main concepts of Business Process Management and focus on the presentation of aspects referring to the modelling of processes. Section 2.2 introduces the main concepts and definitions related to imperative modelling. Section 2.3 gives an introduction of the most important declarative languages characteristics and the study of the most relevant languages. Section 2.4 presents the main concepts and definitions regarding to Constraint Programming. Section 2.5 gives an introduction to the main concepts related to data-quality management. Section 2.6, the case study used to illustrate the solutions proposed in this thesis dissertation is presented.

Part III, IV, and V: Contributions. These parts are the core of the thesis which is comprised of four chapters. The chapters are structured in the following sections: a introduction to the context and motivation for the contribution; a body where the proposal solutions are presented; the results obtained by the application to examples and case of studies; a related work section where a literature review and comparisons with other approaches are given; and to conclude, a summary with a discussion of the results. Specifically, in Chapter 3, a contribution for the specification of the data-oriented optimization problems in business process is presented. Chapters 4 and 5 detail and develop the necessary transformations to make the declarative specification transformed it into an imperative model. In Chapter 6, a contribution for the inclusion of data-quality requirements in the business process models is given.

Part VI: Conclusions and Future Work. This part concludes the thesis over two chapters: Chapter 7, which presents a global summary of the main conclusions that were obtained during the thesis; and Chapter 8, which outlines research lines and topics for future work that may be addressed.

Part VII: Appendices. This part presents the annexes generated as a complement of the information given in the various chapters of the thesis.

Part II
Foundations

Chapter 2

Foundations

Information is knowledge.
Albert Einstein.

The current thesis dissertation aims to improve the existing business processes specifications in order to provide the necessary mechanisms to support data-oriented optimization problems. Therefore, this chapter provides the basis and background regarding business process management, imperative descriptions, and declarative descriptions. Most of the solutions proposed to solve data-oriented optimization problems in business processes are built on the basis of Constraint Programming foundations. Accordingly, the main concepts of Constraint Programming are also detailed. On the other hand, as soon as the data handle in a business process fits a set of data-quality requirements, the optimized product obtained will better fit customers' requirements. To this end, certain concepts related to quality and data-quality management are studied. Finally, the cases study used during this thesis dissertation is also introduced.

2.1 Business Process Management

2.1.1 Concepts

In general, a process can be defined as a set of activities where various organization collaborate to achieve a particular goal. Within the business scope, a process can be defined as a set of activities to help this organization to achieve a goal which provides a value for the company. A business process (cf. Definition 2.1.1) is a special type of process that describes the activities of an organization. One of the main objectives of business processes is to coordinate in a model the activities that conform the performance of a company, providing a single point of access.

Definition 2.1.1. A *Business Process (BP)* consists of a set of activities that are performed in coordination in an organizational and technical environment. These activities jointly realize a business goal (Weske, 2007).

In order to use and manage business processes, business experts need to specify the BPs through BP models (cf. Definition 2.1.2) by using a modelling language. The selection of an adequate graphical method has become an important issue for both academic researchers and business professionals, since each individual process modelling method has its own characteristics. As a consequence, there are many research efforts dedicated to improve the modelling methods. In (Huang et al., 2008), a comparison of these major graphical process modelling methods is presented.

Typically, processes are specified in an imperative way (cf. Section 2.2), i.e., explicitly specifying all possible sequences of activities in a process. However, declarative process models (cf. Section 2.3) have been increasingly used, i.e., implicitly specifying the allowed behaviour of the process with rules that must be followed during execution. Although declarative descriptions are more flexible, since they enable specification of what has to be done instead of specification of how it has to be done, imperative model are easier to understand (Weber et al., 2009).

Definition 2.1.2. A *business process model* consists of capturing which activities, events and states constitute the underlying business process (Weske, 2007). Specifically, the set of activities and the execution constraints between them.

The modelling of the processes plays an important role in the overall management of BPs. In recent years, Business Process Management (BPM) (cf. Definition 2.1.3) has evolved as keystone of in the IT industry. BPM has emerged as an evolution of the traditional Workflow Management (WfM) (van der Aalst, 2004). Moreover, BPM is continuously evolving in order to improve the quality and efficiency of BPs.

Definition 2.1.3. *Business Process Management (BPM)* is an approach that includes concepts, methods, and techniques to support the design, administration, configuration, enactment, and analysis of business processes (van der Aalst et al., 2003; Weske, 2007).

Traditionally, BPs were carried out manually based on the staff knowledge, company regulations and the resources that were already available in the company. Currently, companies can get added benefits if they used software systems to coordinate the activities involved in the BPs. BPM allows organizations to ensure that BPs are executed efficiently, and generate information that can then be used to improve them. It is through the information that is obtained from the daily execution of processes, where potential inefficiencies can be identify, and then, act to optimize them. To this end, it is necessary to have the software (cf. Definition 2.1.4) that provides the necessary support to BPM.

Definition 2.1.4. A *Business Process Management System (BPMS)* is a generic software system that is driven by explicit process representations to coordinate the enactment of business processes (Weske, 2007).

2.1.2 Business Process Management Life-cycle

BPM is orchestrated through a life-cycle such as shown in Figure 2.1. The life-cycle of BPM to support BPs has four phases:

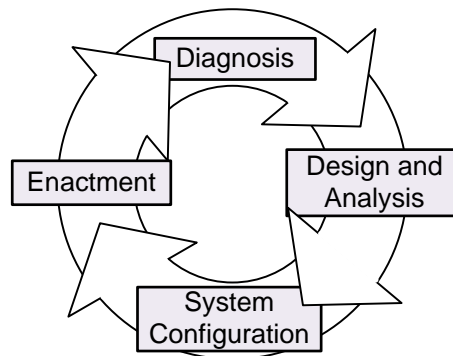


Figure 2.1 – BPM Life-cycle.

- (1) The requirements analysis is established, the business processes are identified, reviewed, validated and presented as process models in the process *design and analysis* phase.
- (2) The designs are developed and configured in a software system in the *system configuration* phase.
- (3) During the process *enactment* phase, the process is executed by using the *system configuration* in the way prescribed by the process model. More specifically, an instance of a BP represents a specific case in the operational business execution of an organization.
- (4) Finally, in the *diagnosis* phase, the operational process is analysed to identify problems in order to improve the process, and can even make a diagnosis with the aim of proposing a solution to these problems.

BPM life-cycle is focused on the design of BP models, and next diagnosis of errors in the execution of these BPs. The creation of complete business process models is a fundamental prerequisite for organizations to complete successfully the life-cycle and to engage the model in BPMS.

2.1.3 Business Process Perspectives

According to Weske, (Weske, 2007), there are two main perspectives in the development of business process in BPM: (1) Operational business process, which defines the activities and their relationships, but implementation aspects of the business process are overlooked; and (2) Implemented business process, which retains information of the execution of activities, technical, and organizational environment in which they will be executed.

Operational business processes are specified by business process models (cf. Definition 2.1.2). In general, business process models must also permit the incorporation of various

perspectives giving place to various diagrams. The diagrams must show the rules, goals, objectives of the business and not only relationships, but also interactions (Castela et al., 2001). A great part of the success of the modelling is the capacity to express the various needs of the business, as well as to have a notation in which these needs can be described. Furthermore, the inclusion of several perspectives in BP models enables to make a more complete and successful execution of BPs. In contrast, the BP models increase the payload work and the complexity of reading comprehension whether the perspective are not clearly separates.

However, although it could not be an easy task, the business process, the environment features, and the intended use of the model must be taken into account to make a successful choice of an approach and/or notation (Bider, 2005). Both, Weske (Weske, 2007) and Van der Aalst (van der Aalst, 2012), differentiate some of the most used perspectives in operational business processes:

- *Functional Perspective*: is the description of the set of activities to be performed in a BP.
- *Control-flow Perspective*: refers to the order in which the activities are performed within a BP. Along with the functional perspective are the base of BP models. This perspective represent the base, which will be enriched with the rest of perspectives.
- *Data-flow Perspective*: includes the set of data used and consumed by the activities during the execution of a BP (Sun et al., 2006).
- *Time Perspective*: refers to the set of temporal constraints to consider during the execution of a BP, e.g. duration of the business process activities, and deadlines constraints (Saoussen Cheikhrouhou and Jmaiel, 2013).
- *Resource Perspective*: focuses on the people, roles, organisational units and any other entities of the organisational model of a company that are involved in a BP (Barba, 2012).

2.1.4 Process-Aware Information System

In order to facilitate the specification and enactment of BPs, Dumas et al. introduced a Process-Aware Information System, henceforth referred to as PAIS (Dumas et al., 2005). PAIS is defined as “*a software system that manages and executes operational processes involving people, applications, and/or information sources on the basis of process models*”. In this way, the PAIS framework and BPs are strongly linked.

As a fundamental characteristic, and opposed to data-centric or function-centric information systems, a PAIS separates process logic from application code (Weber et al., 2009). In addition, Weber et al. provided a PAIS architecture in order to better understand and discuss on the different perspectives of PAIS. PAIS architecture can be viewed as a 4-tier system, as was shown in Figure 2.2, following introduced:

- *Persistence layer* enables the necessary support for a database management system to keep the data persistence.

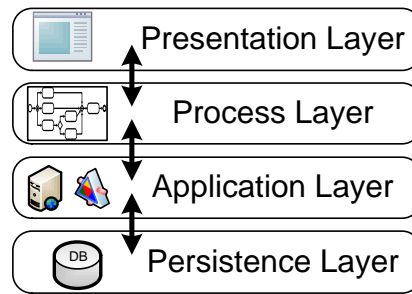


Figure 2.2 – PAIS Framework Architecture.

- *Application Layer* is responsible for storing the application codes and implementations of the different functionalities of the activities. These implementations can be owned by different organizations.
- *Process Layer* runs the process logic. In particular, the schema and complete specification of the process model which is used for the process execution.
- *Presentation Layer* provides different build- and run-time tools to customers, e.g., a process template editor, or an application program interface that enables to monitor the different components.

The different layers are per se four parallel and independent systems, which can be hosted in several machines, running by different and independent applications, and at the same time. This independence is broken from the point of view of the data exchanged, since the different layers are in constant communication exchanging data for providing a given functionality. In addition, the different layers enable that a change, for example in an application service which provides a particular functionality to a process step, will not trigger any other changes in the Process Layer, even it would be possible to state that the interfaces will remain stable (Weber et al., 2009). Currently, the change of the execution order of activities, or the addition of new activities in the Process Layer, can be performed without modifying the implementation of any other application services.

2.2 BP Imperative Description

There are many languages that enable the description of BP in an imperative way. Generally, the common idea of imperative business process modelling is to define a precise activity sequence which establishes how a given set of activities has to be performed. Most commonly imperative languages used for business process modelling and notation techniques are described below. In addition, the standard Business Process Model and Notation, used in this thesis dissertation, is also in-depth described.

2.2.1 Analysis of Imperative Languages

In general, the base of all the imperative languages is the explicit representation of the activities order to be performed, and the relation between them. The main differences are in the way in which they are represented, and the information that can be included in the model. One of the most important assets of imperative languages is the needed effort to transform it into an executable model, the execution engine therefore determines the completeness of an imperative language. Some of the most relevant imperative languages are:

- **Petri Nets** (Petri, 1962) are one of the most known and used techniques to specify business processes in a *formal* and *abstract* way. In addition, Petri Nets establish an important basis for the languages of processes (Weske, 2007). They have a formal and abstract description. *Formal* since the semantics of the instances of the process is well defined and is not ambiguous, and *abstract* since it is independent of the business process execution environment, so that all aspects that are not functional and related to the processes, are not covered (Salimifard and Wright, 2001; van der Aalst and Stahl, 2011). A Petri net is a mathematical representation of a discrete distributed system. They are a generalization of the automata theory that allows expressing concurrent events. A Petri net consists of a finite set of *places* (P), a finite set of *transitions* (T), and a set of directed *arcs* (A). Petri net complies with the characteristics of the bipartite graph: the arcs connect a *place* to a *transition* or a *transition* to a *place*, but cannot be *arcs* between two *places* or two *transitions*. The *transitions* have input and output *places*, representing the input and output source of a *transition*.

The dynamics of systems represented by Petri Nets are modelled with *tokens*. These *tokens* are located in various *places*, which can contain any number of *tokens*. When the structure of Petri Nets is fixed, the *tokens* should change their positions according to the fixed rules. The distribution of the *tokens* between the *places* determines the Petri network status, and the system modelled by it. As the *transitions* can change the status of the Petri network, they can be considered as active components representing events, operations, transformations and transport. A *place* is a passive component, such as a buffer, a state, or a condition. The *tokens*, on the other hand, represent the information or physical objects. In the context of business processes, the *transitions* represent the activities, and the *places* containing the *tokens* represent instances of the process states.

Petri Nets present some limitations for modelling complex BPs, in these cases Workflow nets are used (Salimifard and Wright, 2001). Specifically, Petri Nets require expert knowledge to be used. De Backer et al. in (De Backer and Snoeck, 2005) introduce the application of Petri Net language theory for business process specification. Petri Net languages (Bosilj-Vuksic and Hlupic, 2001) are an extension to the Petri Net theory (Petri, 1962), providing a set of techniques to describe complex business processes more efficiently.

- **Activity Diagram.** In the Unified Modelling Language (UML), an activity dia-

gram represents workflow step of business and operational components in a system (Castela et al., 2001). In UML 1.x, an activity diagram is a variation of the diagram of States UML where the *states* represent operations, and the *transitions* represent the activities that occur when the operation is completed. Activities in UML 2.0 diagram, is similar in appearance to the activities UML 1.x diagram, but they have a semantic based on Petri Nets (List and Korherr, 2005). In UML 2.0, the general diagram of interaction is based on the activity diagram which is focused on the flow of the activities involved in a single process by showing the general control flow of a system, with its activities and actions. Specifically, the OMG specification defines an activity as (OMG, 2005) diagram: “*a variation of a state machine, where the states represent the performance of actions or sub-activities, and the transitions are provoked by the realization of the actions or sub-activities*”. Therefore, the purpose of the activity diagram is the modelling of a process and its operations.

In (Russell et al., 2006), the authors examine the suitability of UML 2.0 Activity diagrams for BP modelling. The pattern evaluation shows that UML 2.0 Activity Diagrams is not suitable for representing all aspects of this type of modelling. It offers support for control-flow and data perspective allowing most of the constructs to be directly captured. However, it is extremely limited in modelling resource-related or organizational aspects of business process. These limitations are shared with most of the other imperative languages, showing the emphasis that has been placed on the control-flow and data perspectives in these notations. Despite its limitations, several authors are still been using UML 2.0 Activity Diagram to model their BP. For example, Rodríguez et al. in (Rodríguez et al., 2006) extends UML 2.0 in order to include many security requirements in the BP models. On the other hand, there are many works that compare the readability, and boundaries of using UML 2.0 Activity Diagrams with regard to other BP modelling languages, such as the most used standard BPMN (Peixoto et al., 2008; Venera Geambasu, 2012).

- **Event-driven Process Chain (EPC)** is a type of flowchart for the modelling of business processes (Tsai et al., 2006). Its main objective is to represent the concepts of domain and processes rather than their formal aspects or technical. EPC is used to configure the implementation of an enterprise resource planning and business process improvement. An EPC is an ordered graph of events and functions (Tsai et al., 2006), so it provides several connectors that allow alternative and parallel execution of processes. Moreover, it is specified by the use of logical operators such as AND, OR and XOR. One of the features that confirms to EPC is an acceptable technique to represent business processes, since its notation is simple and easy to understand. The EPCs are represented by means of directed graphs where a specific order between the nodes is provided. They resemble UML diagrams, since their structures provide trivial structures, by means of introducing restrictions or parallelism in their execution semantics. As counterpart, although it is simple, EPC may be a bit ambiguous since nor its semantics and syntax are completely and well defined. This ambiguity suppose a high level of difficulty to check the consistency and completeness of the models. All these problems are serious since EPC is used

as specification of BP so that they can be processed by systems that support and enable BP. The absence of the formal semantics also impedes the exchange of models between tools from different vendors and avoids the use of powerful analytical techniques.

These problems are discussed in (Van der Aalst, 1999), under an approach based on Petri Nets, explained below. The building blocks used in an EPC (events, functions and connectors) are close to the building blocks used in a network of Petri (places and transitions). In fact, EPC corresponds to a subclass of Petri Nets. Shows that it is possible to assign an EPC in a Petri net. In this way the formalism of Petri Nets can be used to give formal semantics to the EPC. In addition, advanced techniques of Petri Nets can be used to analyze the EPC.

- **Integration DEFinition diagram** is a family of modelling languages in the field of systems and software engineering. They cover a wide range of applications, from functional modelling, simulation, analysis, object-oriented design and acquisition of knowledge. These languages were developed with funding from the U.S. air force, but currently they are in the public domain. From IDEF family, the most widely recognized and used is IDEF0. IDEF0 is a functional modelling language developed in SADT (Structured Analysis and Design Technique). However, the most interesting is the IDEF3 diagram (Bosilj-Vuksic and Hlupic, 2001). The IDEF3 models the process flows, which describes how activities work together to form a process. The IDEF3 diagram identifies the behaviour of the system by describing the structured of *what* the system really does and *how* the activities work together to form a process. There are two types of description: *Flowchart* of the process, and the *state network* of the object transitions.
- **Process Specification Language (PSL)**, (NIST, 2004), is a standard language for process specification that serves as an interlingua to integrate multiple process-related applications throughout the manufacturing life cycle. The main goal of PSL is to create a process representation that is common to all manufacturing applications: generic enough to be decoupled from any given application, and robust enough to represent the necessary process information for any given application.
- **Business Process Model and Notation (BPMN)** is the standard proposed by OMG (OMG, 2011b). BPMN is the most used language in the market for the business process modelling. BPMN is a graphical notation with which enables the creation of various diagrams within the three types of sub-models (private, public and collaborative). BPMN diagrams is formed by a set of elements detailed in the following subsection 2.2.2.
- Another important imperative BP specification language is **XML Process Definition Language (XPDL)** (WfMC, 2004). In this case, XPDL provides an XML file format that can be used to interchange process models between tools. This language is designed to exchange the process definition, through both the graphics and the semantics of the workflow in a business process. Nevertheless, it has been

designed specifically to store all aspects of a Business Process Model and Notation (BPMN) diagram (OMG, 2011b). Business Process Model and Notation provides graphical notation to facilitate human communication between business users and technical users, of complex business processes. Thus, XPDL and BPMN specifications address the same modelling problem from different perspectives. As occurs with BPMN, the remaining imperative languages describe a BP graphically and formally.

In (Huang et al., 2008; List and Korherr, 2006), a comparison of these major graphical process modelling languages is presented. Extensive literature research, regarding another type of business process compliance, has been also presented in (Namiri and Stojanovic, 2007; Sadiq et al., 2007). Although it is possible to find tools that execute the model represented by the various languages explained, commercial tools are focused especially to support the execution of BPMN. Although, several BPMN engines develop their own executable platform with proprietary code, one of the most widely used standard executable languages is the Business Process Execution Language (BPEL) (OASIS, 2007). The BPEL enables actions within a BP to be specified with web services. A BPEL specification, once written, can be compiled into executable code that implements the described BP. In addition, BPMN specification also provides a mapping between its notation and BPEL notation.

The use of imperative models in the software industry has been prevailing, since there exist an import number of commercial tools to support them, such as IntalioTM (Community, 2012b), ActivitiTM (Team, 2012), and Bonita Open SolutionTM (Community, 2012a). Unfortunately, they lack in flexibility, expressiveness and adaptability, being necessary more declarative languages.

2.2.2 Business Process Model and Notation (BPMN)

For the performing of the current Thesis Dissertation, BPMN 2.0 (OMG, 2011b) has been chosen to introduce our proposal. BPMN is an international standard for process modelling accepted by the community, which is independent of any process modelling methodology. It is specifically designed to coordinate the sequence of the processes and the messages flowing between participants in the different activities. BPMN provides a common language to enable the communication between processes in a clear, complete, and efficient way.

Business process models specify the activities, with their relationships, that are performed within a single organization, or between activities of different processes participating in a business-to-business collaboration. The interaction between activities in different processes is just through sending and receiving messages. However, within each single business process, the decisions and branching of flows are modelled using gateways, also called control flow patterns.

BPMN is a graphical notation which enables the creation of various diagrams within the three types of sub-models (private, public and collaborative). BPMN diagrams is formed by a set of elements (OMG, 2011b) that can be grouped in:

1. **Sequence Elements** are the fundamental building block for workflow processes. It represents a series of activities which are executed in turn one after the other.
2. **Connectors** connect objects in the flow, pools or artefacts.
3. **Pools and Lanes** represent the organization aspects of business process diagrams.
4. **Pools and Lanes** represent the organization aspects of business process diagrams
5. **Artefacts** show additional information about the business processes that are not sufficiently relevant to be included in the sequence flow or the message of the process

Although the specification and details of every elements can be found in the document associated to the standard (OMG, 2011b), Table 2.1 describes the main elements used in this thesis dissertation. There are different variants of these elements, but which are not described, since they are not going to be used in this work.

Table 2.1 – Main BPMN Elements used in this thesis dissertation

Type	Description
<i>Sequence Elements</i>	
Events	Something that happens during the course of a BP. They can be of three types: beginning, intermediate and end.
Activity	The generic term to describe any work performed by the company. They may be atomic or compound.
Gateway	To control the flow, it can be a traditional decision, a join, a merge, or a fork.
<i>Connector Elements</i>	
Sequence Flow	Defines the execution order of activities.
Message Flow	Symbolizes information flow across organizational boundaries.
<i>Pool Element</i>	
Pool	It enables to indicate the participants in the process.
Lane	A participant of the POOL. It represents responsibilities of activities in the pools.
<i>Artefacts Element</i>	
Data Object	Enables to show the data that are produced or required by the activities
Data Input	External input for the entire process
Data Output	Variable available as result of the entire process

Although BPMN 2.0 solves the majority of the modelling problems related to the combination of activities, by means of conversations and choreographies, it remains as yet insufficiently powerful to represent the combination of activities oriented to the optimization of the outcome data. For this reason, one of the proposals presented in Chapter 3 is the extension of the expressiveness of BPMN 2.0 with a new type of sub-process.

2.3 BP Declarative Description

As was commented in previous chapters, imperative models lack in expressiveness, flexibility and adaptability. One of the reason is that sometimes is easier to describe what the process does, rather than how the process achieves it. In order to solve this lack, there are many languages that enable the description of BP in a declarative way. Generally, the common idea of declarative business process modelling is to model a process as a trajectory in a state space. Moreover, declarative constraints are used to define the valid movements in that state space (Bider et al., 2000). The differences between declarative process languages can, in part, be understood as a different perception of the meaning of 'state'.

One of the main aspects to consider in a data-oriented optimization process is the simplicity to be described in a declarative way, since the set of activities, the constraints that relate their data, and the objective function are the main BP requirements. However, an imperative specification is not so simple with this information since the model possibilities are too long to deal with. Therefore, we have found some characteristics that we consider interesting to be analysed in the existing declarative languages in order to know if they enables to describe and support data-oriented optimization problems in BPs. The characteristics are studied in Subsection 2.3.1, the existing proposals are analysed in Subsection 2.3.2 and then compared in Subsection 2.3.3.

2.3.1 Characteristics

The characteristics presented in this section are oriented towards the subsequent analysis of the different proposals related to declarative languages. The characteristics serve as a base to determine whether the languages address the data management, have the necessary mechanisms for reasoning about the components declared, and/or include the BP requirements necessary to specify a data-oriented optimization process.

- **Formalism for reasoning:** the proposals use different formalism for reasoning. Sometimes, although we show the most relevant in each case, they can combine more than one, and/or be improved by means of made-to-measure algorithms. In order to centre the attention, only the most relevant have been included.
 - **Linear Temporal Logic (LTL).** LTL is a modal temporal logic that allows the expression of temporal constraints on infinite paths within a state space. As demonstrated by Chomicki (Chomicki, 1995), Bacchus and Kabanza (Bacchus and Kabanza, 2000), and Pesic and van der Aalst (Pesic and van der Aalst, 2006), LTL expressions can be used to represent desirable or undesirable patterns within a history of events. LTL formula can be evaluated by obtaining the Büchi automaton (Büchi, 1990) that is equivalent to the formula, and checks whether a path corresponds to the automaton. LTL was spatially used to verify or monitor running programs based on the analysis of events, but in BPM fields, LTL has also be used to represent graphically the declarative models (Maggi et al., 2011a). Unfortunately most LTL checking algorithms assume

infinite paths and construct non-deterministic automata (Pesic and van der Aalst, 2006). Another disadvantage is that LTL does not allow the expression of the effect that results from a particular transition in a state space. For these reasons, it is not evident to express a goal state in LTL nor to construct automata for planning an execution scenario to obtain a goal state (Bacchus and Kabanza, 2000) as is needed in an optimization function.

- **The Event Calculus.** In first-order logic, there is a formalism that elegantly captures the time-varying nature of facts, the events that have taken place at given time points, and the effect of these events reflecting on the state of the system. This formalism is called the Event Calculus. The Event Calculus, introduced by Kowalski and Sergot (Kowalski and Sergot, 1986), is a logic programming formalism to represent and reason about the effect of events on the state of a system. In addition, the Event Calculus not only has the ability to deductively reasoning about the effects of the occurrence of events (leading to the coming into existence of fluency or the ceasing to hold), most important is that it has also the ability to reason abductively. Abductive reasoning over the event calculus has been demonstrated to be equivalent to planning. In particular, abductive reasoning produces a sequence of transitions (denoted by events) that must happen for a particular instance to hold in the future (Eshghi, 1988), (Nuffelen and Kakas, 2001) and (Shanahan, 1997).
- **Coloured Petri Nets (CPNs)** (Bosilj-Vuksic and Hlupic, 2001; van der Aalst and Stahl, 2011) is a backward compatible extension of Petri Net (cf. Section 2.2.1). CPNs preserve useful properties of Petri Nets and at the same time extend initial formalism to allow the distinction between tokens. Specifically, CPN is a graphical language for constructing models of concurrent systems and analysing their properties, thereby including the capabilities of a high-level programming language. Petri Nets provide the foundation of the graphical notation and the basic primitives for modelling concurrency, communication, and synchronisation, for many complex applications, a more natural and compact description is possible if tokens carry information, being necessary to use CPNs.
- **Graph theory.** The business process can be modelled as a directed labelled graph, for this reason sometimes authors have based their business processes representation and the evaluation of them in the use of graph theory, and including some made-to-measure algorithms (Becker et al., 2013; Fan and Weinstein, 1999).
- **Pattern Matching.** Generally, a pattern is a plan, diagram or structure used as a guide in making something. Applied to business process, patterns are mainly applied in the modelling stage. Similar to design patterns in software development, a business process pattern describes a design solution to an operational business problem. The patterns can be used to model the BP (Gschwind et al., 2008), suggest additional actions for a process model (Smirnov et al., 2009), or the monitoring of BP compliance (Ly et al., 2011). Therefore, the

base of pattern matching consists of finding the similarities of the model wanted and/or structure found to some existing pattern in order to reuse and recognise known pre-design models and situations.

- **Constraint Programming (CP).** Constraint Programming (Rossi et al., 2006) is a paradigm that permits the declarative description of a model by means of constraints and variables. These constraints and variables enables to represent the relation between the components of a model, both the time order execution and data values that flow between activities. As this is the proposal that we use in the thesis, we describe it in-depth way in Section 2.4.1.
- **Imperative and Declarative:** the capacity of a language to describe imperative and declarative aspects in the same model. Sometimes, a part of the process is completely unknown, which could be described declaratively, and other parts are totally known, which could be described imperatively. There exists proposals that can combine both types of description in the same model.
- **Use of the model:** the existing proposals that we have analysed are focused on different objectives: *Validation* of the model at runtime for a trace of events, *Construction* of automatons to generate a possible sequence of activities to simulate possible traces, or *Assistance* to the customer to decide which is the best activity to execute at runtime to compliance the model.
- **Data perspective:** some of the languages give the possibility to include the values of the data-flow variables in the rules that describe the declarative model, not restricting the description to the activities order.
- **Pre and Post-condition:** derived from the declarative description of the model, sometimes is interesting to describe how is the system before and after the process is executed. It can be done by means of pre and post-conditions. This is a relevant aspect, especially in data-aware processes, since it allows the modeller to describe the data before and after the process execution. This avoids the inclusion of details about how to obtain the post in function of the pre-condition, since these details belong to the internal description of the process.
- **Optimization Function:** the declarative models are usually used when several possible sequence of activities can be executed, being not possible to determine a explicit combination. Frequently, to select one of the correct possibilities can be better or worse depending on the spent time, or resources and data used during the execution. Some proposals permit to include an objectives to minimize or maximize in order to opt for an instance from another. The possibility to include an optimization function in the declarative description is an important characteristic to take into account.

2.3.2 Analysis of Declarative languages

Several declarative languages can be found in the literature, some of the most important have been included in this section.

- **Pocket of flexibility.** This solution is based on constraint specification of the business process workflow. The constraint specification framework (Sadiq et al., 2005) represents the workflow as a directed graph where there are two types of nodes: activity nodes and coordinator nodes. In the framework, it is possible to combine activities whose relation is known with activities whose relation is unknown (called pocket of flexibility). The framework includes a set of constraints for concretizing the pocket with a valid composition of workflow fragments. It also includes different types of order constraints (Serial, Order, Fork, Inclusion, Exclusion, and Parallel). The constraints also describe the number of times that each activity can be executed, and whether the execution of each activity is mandatory. The proposal defines a set of algorithms to find possible discrepancies between the constraints that describe the process and an instance at runtime. The implementation is based on a made-to-measure algorithm that uses the graph to represent the constraints. The implementation has been included in the prototype called *Chameleon*. The data aspect nor the optimization objective have not been included in this proposal.
- **DeCo.** Irina Rychkova et al. in (Rychkova and Nurcan, 2011; Rychkova et al., 2008a,b) presented a declarative BP specification language that enables designers to describe the actions that a business process needs to contain, but not where their specific sequence can be postponed to the instance time. They improve the alignment of the BP with the business strategy of an organization by giving a synthesis of a set of business processes (abstracting the control flow), while maintaining a rigorous relationship with the detailed process. These specifications complement the traditional (imperative) business process model by specifying the process independently from a particular environment where the process can be executed. This proposal includes checking the conformance of the imperative and the declarative specifications, using the case handling paradigm (van der Aalst et al., 2005). For every action of the working object, they define a pre-condition and a post-condition. A precondition specifies a set of states where the action can be executed, and post-condition specifies the possible set of states after the execution of the action. The pre and postcondition represent how the different actions can modify the state of the objects transformed during the process execution, they do not define the order of the actions. Therefore, different imperative description for the same declarative descriptions are possible. Thereby this proposal focuses on the problem from the working object point of view, and data values is one of the analysis.
- **Compliance Rule Graphs.** The Compliance Rule Graphs (CRGs) (Knuplesch et al., 2010; Ly et al., 2008, 2011) focus their challenge on finding an appropriate balance between expressiveness, formal foundation, and efficient analysis. For these reasons, the authors propose a language based on a graph representation where the order of the activities and the occurrence or absence of activities can be included as

well. The proposal verifies the correctness of the process analysing the compliance rules and the events monitored. The description of the order of activities can be enriched including conditions to the rules, that will be satisfied or not depending on the data value for each instance. The analysis is done using pattern matching mechanisms, and is included in a prototype called SeaFlow (Ly et al., 2010).

- **Em-Bra²Ce.** The Enterprise Modeling using Business Rules, Agents, Activities, Concepts and Events (Em-Bra²Ce) Framework (Goedertier and Vanthienen, 2007; Roover et al., 2011) presents a declarative language based on the standard SBVR (Semantics Of Business Vocabulary And Business Rules) (OMG, 2008) to describe the vocabulary of the process, and an execution model to represent the control flow perspective based on Colored Petri Nets. The use of SBVR allows the description of data aspects in the business process that can be included in the Event Condition Action rules, used as a pattern to write the rules.
- **Penelope.** The language Penelope (Process ENtailment from the ELicitation of Obligations and PERmissions) (Goedertier and Vanthienen, 2006) expresses temporal rules about the obligations and permissions in a business interaction using Deontic logic. This language is supported by an algorithm to generate compliant sequence-flow-based process models that can be used in business process design. This language uses the Event Calculus to model the effects of performing activities with respect to the coming into existence of temporal deontic assignments. The only type of data that is included in the definition is related to the execution time of the activities, but the data managed during each instance is not an object of the proposal.
- **ConDec.** The ConDec (Pesic and van der Aalst, 2006) language was designed for modelling and enacting dynamic business processes. The language defines the involved activities in the process and the order relations between them. This order relation is expressed using LTL to represent desirable or undesirable patterns within a history of events. However, LTL formulas are difficult to read due to the complexity of expressions. Therefore, the authors have defined a graphical syntax for some typical constraints that can be encountered in workflows. ConDec initially defined three groups of templates to make the definition of activity relations easier: (1) existence, (2) relation and (3) negation templates. An automaton can be built in accordance with the ConDec model, where the automaton can be used to validate a sequence of events. Declare tool (Maggi et al., 2014) is a prototype of a workflow management, that supports the ConDec language. This tool has been used for frameworks such as *Mobucon* (Maggi et al., 2011a,b) for runtime validation. This framework allows the continuous verification of compliance with respect to a predefined constraint model. ConDec has been enlarged to include the resource perspective (ConDec-R) and the data-aware constraints in Declare, both analysed in the following items.
- **ConDec-R.** This is an extension of the ConDec language to include a description of the resources necessary during process execution. The implementation extension,

called ConDec-R (Barba et al., 2013), assists the customer by means of recommendations to achieve an optimized plan for one or multiple objectives (Jiménez Ramírez et al., 2013). In order to obtain the plan, a CP paradigm is used, combined with a set of algorithms to minimize evaluation time. Although this proposal incorporates the resource perspective which is a type of data, this type of information is not oriented to activity input and output data.

- **Data-aware Constraints in Declare.** This is an extension of the Declare framework (Montali et al., 2013) that permits the representation of the input, internal and output data of the activities in a declarative representation of the order of the activity. Event calculus has been used to formalize the language and to validate if the traces of events are in accordance with the declarative model. Although the data aspect is included, only input and output data relations between activities can be described.

2.3.3 Comparative

Although all these declarative languages include some information about data, none of them include the data input and output of the activities with the aim to optimize the object obtained from the business process. As shown in Table 2.2, none of the declarative language include all these characteristics in their models, allowing only some of them. Each declarative language used a different formalism of reasoning, since each one enables to obtain a control-flow sequence based on the declarative components specified in the model. Half of the proposals enable to specified imperative and declarative components in their models. This characteristic use to be desirable whether not always a sequence of activities is clearly known at design time, being easily to describe its requirements through declarative components. On the other hand, most of them use the declarative models to validate if a sequence of activities, for a given instance is correct or not based on the BP requirements. But not take advantages to build a generic process capable of support every instance or assist the customers at runtime. Related to the data perspective, although most of them include some information about, none of them include the data input and output of the activities with the aim to optimize the object obtained from the BP. However, it does not occur for the specification of the pre- and post-conditions of the process, including only by DeCo. Although the objective function is only considered by ConDec-R, it is focused on the optimization of time and resources, not on the outcome data of the BP to be offered to customers.

Furthermore, it is necessary to highlight that these declarative languages are not worried to support the execution of any instance at runtime as occurs in the imperative languages. They only provides tools to guide and recommend how a concrete instance must be executed.

Table 2.2 – Declarative Languages Comparative

Language	Formalism	Imper. and Decl.	Use of model	Data persp.	Pre and Post	Opt. Funct.
Pocket of Flex.	Graph theory + algorithm	✓	Val.			
DeCo	First Order Logic	✓	Val.	✓	✓	
CRGs	Pattern matching		Val.	✓		
Em-Bra ² Ce	Color Petri Net	✓	Val.	✓		
Penelope	Event calculus	✓	Constr.			
ConDec	Linear Temporal Logic		Val.			
ConDec-R	Constraint Programming		Assist.	✓		✓
Data-aware	Event Calculus		Val.	✓		

2.4 Constraint Programming Paradigm

The transformation of the declarative model into an imperative model entails the analysis of every possibilities to find the optimal one. In order to realize this process, we have decided to use Constraint Programming paradigm since it is a declarative model that permit to find a solution that compliance with this analysis. The basic concepts and search algorithms used in Constraint Programming paradigm are explain in following subsections.

2.4.1 Constraint Programming Concepts

Constraint Programming is based on the resolution of Constraint Satisfaction Problems (CSPs), which are problems where an assignment of values to variables must be found in order to satisfy a number of constraints. A large number of problems in Artificial Intelligence and other areas of Computer Science can be seen as special cases of CSPs. Examples include scheduling, temporal reasoning, graph problems, configuration problems, etc.

In general, a CSP (cf. Definition 2.4.1) is composed of a set of variables, a domain for each variable, and a set of constraints. Each constraint is defined over some subset of the original set of variables and limits the combinations of values that the variables in this subset can take. The goal is to find one assignment to the variables such that the assignment satisfies all the constraints. In some kind of problems, the goal is to find all such assignments (Kumar, 1992).

Definition 2.4.1. A *Constraint Satisfaction Problem (CSP)* consists of the triple (V, D, C) , where V is a set of n variables v_1, v_2, \dots, v_n whose values are taken from finite, discrete domains $D_{v_1}, D_{v_2}, \dots, D_{v_n}$ respectively, and C is a set of constraints on their values. The constraint $c_k(x_{k_1}, \dots, x_{k_n})$ is a predicate that is defined on the Cartesian product $D_{k_1} \times \dots \times D_{k_n}$. This predicate is true iff the value assignment of these variables satisfies the constraint c_k .

The search of solutions for a CSP is based on the instantiation concept. An assignment of a variable, or instantiation, is a pair variable-value (x, a) which represents the

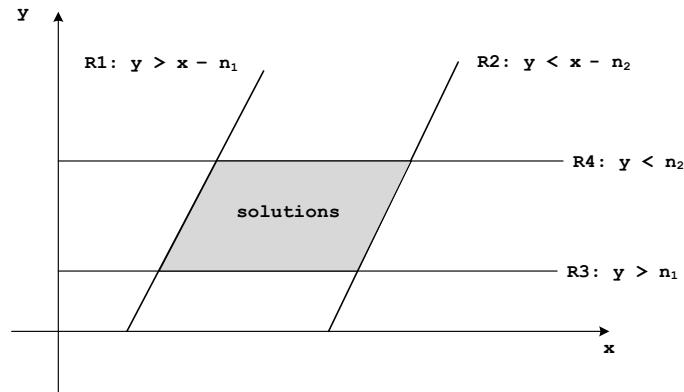


Figure 2.3 – Graphical solution of a CSP system

assignment of the value a to the variable x . An instantiation of a set of variables is a tuple of ordered pairs, where each sorted pair (x, a) assigns the value a to the variable x . A tuple $((x_1, a_1), \dots, (x_i, a_i))$ is consistent if it satisfies all the constraints formed by variables of the tuple.

A solution to a CSP is an assignment of values to all the variables that all constraints must be satisfied. Hence a **solution** is a consistent tuple which contains all the variables of the problem. A partial solution is a consistent tuple which contains some of the variables of the problem. A problem is consistent if it exists, at least, a solution, i.e., a consistent tuple. In Figure 2.3, there is a graphic which represents the space of solutions of a CSP that must satisfy four constraints ($R1$, $R2$, $R3$, and $R4$) therefore the space of solutions is restricted to the grey-highlighted rectangle.

2.4.1.1 CSP Consistency

One of the main difficulties in CSP resolution is the appearance of local inconsistencies. Local inconsistencies are values of the variables that cannot take part of the solution because they do not satisfy any consistency property. Therefore if any consistency property is forced, we can remove all the values which are inconsistent in regard to the property. But it can be possible that some values which are consistent in regard to a property are inconsistent in regard to another property at the same time. Global consistency implies that all values which cannot take part in a solution can be removed. The constraints of a CSP generate local inconsistencies because they are combined. If the search algorithm does not store these inconsistencies, it will waste time and effort trying to carry out instantiations which have already been tested.

2.4.1.2 CSP Search Algorithms

Various approaches to solve CSPs have been developed, a number of them which use constraint propagation to simplify the original problem. Others use backtracking to directly

search for possible solutions. Several are a combination of these two techniques.

The techniques used in constraint satisfaction depend on the kind of constraints being considered. Constraints are often used on a finite domain, to the point that CSPs are typically identified with problems based on constraints on a finite domain. Such problems are usually solved via techniques that combines propagation and searches, in a particular form of backtracking and local searcher. Constraint propagation is another method used on such problems; the majority of them are incomplete. In general, they may solve the problem or prove it unsatisfiable, but not always. Constraint propagation methods are also used in conjunction with searches to make a given problem simpler to solve. Other considered kinds of constraints are on real or rational numbers; solving problems on these constraints is done via variable elimination or the simplex algorithm (Marriott and Stuckey, 1998).

The search techniques to find solutions to a CSP are based normally on search algorithms such as backtracking or exhaustive. They try to find a solution through the space of possible assignments of values to the variables, if it exists, or to prove that the problem has not a solution. Because of this, they are known as complete algorithm. The incomplete algorithms such as local searches do not guarantee to find a solution, but they are very used in optimization problems since their mayor efficiency and the high cost that a complete search requires. A lot of complete search algorithms have been developed.

When solving a CSP, it is necessary to assign values to variables satisfying a set of constraints. In real applications it often happens that problems are over-constrained and do not have any solution. In order to solve these problems, several extensions of the model have been proposed, where it is allowed to contain weak constraints (which indicate preferences, not obligation) with different semantics, such as priorities, preferences, costs, or probabilities.

2.4.2 Constraint Optimization Problems

Sometimes, the problems are not only interested in the satisfiability of a set of constraints but also want to find the “best” solution to the constraint. There are often a lot of solutions to a CSP, which means that a user is interested only in some of them, or only in a specific one. In general these solutions reduce the space of solutions to a sub-set of the solutions. Finding a “best” solution to a constraints is called an *optimization problem* (Marriott and Stuckey, 1998). This requires some way of specifying which solutions are better than others. The usual way of doing this is by giving an objective function that has to be optimized.

Definition 2.4.2. A *Constraint Optimization Problem (COP)* consists of the tuple (V, D, C, O) , where an objective function O is included in a CSP defined by the tuple V, D, C . The objective function implies maximizing or minimizing a variable that can represent a numerical combination of others by means of a function.

Optimization problems do not necessarily have a single optimal solution. For example, consider the constraint $X + Y \leq 4$ together with the objective function $X + Y$. Any solution of the constraint $X + Y = 4$ is an optimal solution to this optimization problem.

2.4.2.1 COP Search Algorithms

As occurs with CSP search algorithms, the methods used to solve optimization problems depend on specific problem types. Optimization problems can be categorized according to several criteria. Depending on the type of functions involved there are linear and nonlinear (polynomial, algebraic, transcendental, ...) optimization problems. We could use a solver to find any solution to the CSP, and then add a constraint to the problem which excludes solutions that are not better than this solution. The new CSP is solved recursively, giving rise to a solution which is closer to the optimum. This process can be repeated until the augmented CSP is unsatisfied, in which case the optimal solution is the last solution found.

One of the algorithms most widely used in practice is Dantzig's simplex algorithm (Marriott and Stuckey, 1998). However, constraint optimization can be solved by branch and bound algorithms, which are better known and whose use is more common. These are backtracking algorithms storing the best solution found during execution and use it for avoiding part of the search. More precisely, whenever the algorithm encounters a partial solution that cannot be extended to form a best solution than the stored, the algorithm backtracks, instead of trying to extend this solution. The efficiency of these algorithms depends on how the best solution that can be obtained from extending a partial solution is evaluated. Indeed, if the algorithm can backtrack from a partial solution, part of the search is skipped.

2.4.3 Distributed Constraint Satisfaction Problem

In (Yokoo et al., 1998), Yokoo et al. presented a Distributed Constraint Satisfaction Problem (DisCSP) as a general formalism for dealing with problems in multi-agent systems. A DisCSP can be considered as a CSP in which variables and constraints are distributed among multiple agents and the agents are required to satisfy all constraints by communicating with each other. In other words, a DisCSP is a CSP where the set of variables and constraints of the problem are distributed between a set of agents who are in charge to solve their own sub-problem and must coordinate themselves with the rest of agents to reach a solution to the global problem (Abril López et al., 2007).

In (Hirayama and Yokoo, 1997), the authors define DisCSP as:

- A set of agents, $1, 2, \dots, m$.
- A set of n variables $V = x_1, x_2, \dots, x_n$ where the values of the variables are taken from finite, discrete domains D_1, D_2, \dots, D_n , respectively
- For each variable x_j , an agent i is defined such that x_j belongs to i . We mean x_j belongs to i by belongs (x_j, i)
- A constraint C_l is known by an agent i . The predicate known (C_l, i) is used to express that.

Only the agent who is assigned a variable has control of its value and knowledge of its domain. The goal for the agents is to choose values for variables such that a given global objective function.

We assume, in general, that an agent knows only those constraints relevant to the variables that belong to it. Note that some constraints known by an agent may include other agents' variables, not just its own variables. We refer to such a constraint as an inter-agent constraint.

A distributed CSP is solved when the following conditions are satisfied for all agents. For each agent i ,

- A variable x_j has a value d_j as its assignment for $\forall x_j$ belongs (x_j, i) .
- A constraint C_l is true under the above assignments for $\forall C_l$ known (C_l, i) .

In addition, the combination of web services can also implied to obtain the best solution or one of the best. It provokes to build and solve a Distributed Constraint Optimization Problem (DisCOP) (Silaghi and Yokoo, 2009).

2.4.3.1 Algorithms for solving DisCSP

The algorithms for distributed CSPs must find a solution as quickly as possible. An agent in a distributed CSP has only limited knowledge of the entire problem, and thus important things for the algorithms include how agents communicate with each other and what information is transferred. There are several proposals focused on solve DisCSPs. The majority of these studies aim at defining distributed backtracking-based search algorithms, therefore some of them are studied in the following.

- **Centralized Backtracking** selects a leader agent among all agents who gather all the information about variables, their domains, and their constraints (Yokoo and Hirayama, 2000). If the knowledge about the problem can be gathered into a single agent, this agent can solve the problem alone by means of the utilization of normal centralized constraint satisfaction algorithms.
- **Synchronous Backtracking** assumes that the agents agree on an instantiation order of their variables (such as agent x_1 goes first, then the agent x_2 , and so on). Each agent, receiving a partial solution (the instantiations of the preceding variables) from the previous agent, instantiates its variable based on the constraints that it knows about. If it finds a value, it will append this value to a partial solution and will pass it on to the next agent. If no instantiation of its variable can be satisfy the constraints, then it sends a *backtracking* message to the previous agent (Yokoo et al., 1998).
- **Asynchronous algorithms (ABT)** are characterized by the fact that all agents are active in parallel and only coordinate as needed to ensure consistency if their variables are involved in the constraints.

In Asynchronous Backtracking the priority order of agents is determined, and each agent communicates its tentative value assignment to its neighboring agent via 'ok?' messages. Each agent maintains the current value assignment of other agents from its viewpoint. An agent changes its assignment if its current value assignment is not

consistent with the assignment of higher priority agents. If there exists no value that is consistent with the higher priority agents, the agent generates a new constraint (called a *nogood*), and communicates the *nogood* to a higher priority agent, thus the higher priority agent changes its value making backtracking.

In (Yokoo and Hirayama, 2000) the soundness, completeness and termination of ABT algorithm is proved.

On the other hand, in order to improve the time to find the best solution, the paper of Redouane Ezzahir et al. (Ezzahir et al., 2008) proposes an algorithm for solving Distributed Constraint Optimization Problems. The algorithm is based on branch and bound search with dynamic ordering of agents. This algorithm can be adapted to find the *nogood* values in a cooperative space, helps pruning dynamically unfeasible sub-problems and speeds up the search.

2.5 Data Quality Management in Business Process

The manage of data in business process also implies to be worried about the level of quality of the data handle, since a business process cannot work correctly with low-level quality of the data. For this reason, in this section the principal aspect of data quality management and how they can affect to a business process are analysed.

2.5.1 Data Quality Management Concepts

In spite of the widest usage of the classic definition of **data quality** (DQ) as *fitness for use* (Wang, 1998), in this thesis dissertation we rather prefer the definition of *meeting requirements* (Crosby, 1979) because it brings certain advantages from the point of view of implementing the various mechanisms of the DQ Layer. In addition, this definition also involves the barely used concept of *internal data quality* vs *external data quality* as a way in which a set of data satisfies the stated requirements. This concept forces differences to be highlighted between: (i) what defines the data (as a product); and (ii) what factors are specific to the assessment of the quality. The reason for such distinction is to better identify and separate certain aspects of the quality of “*data product*”, aspects that may or not fit with what defines the data. The data is defined by means of certain features and sub-feature (cf. Definition 2.5.1 and Definition 2.5.2) that are set up through the design process in accordance with the data requirements. Therefore, in the case when data fails to satisfy the requirements, this will be due to the fact that corresponding features are not properly designed or used.

Definition 2.5.1. Feature: any of the components of the piece of data: name, attribute, value, data type.

Definition 2.5.2. Sub-feature: any of the sub-components of a feature: e.g. data type could be decomposed into the specific data type (numeric, character,...) and size.

When the assessment of the level of quality of a piece of data is required, the specification of a number of **DQ requirements** against which DQ is to be judged become necessary. These DQ requirements can be of two types: **High-Level Data-Quality Requirement**, such as control and assurance; and **Low-Level Data-Quality Requirements**, such as measurement, assessment and decision-taking (Cappiello et al., 2013). For the definition of high-level DQ requirements, some criteria, commonly known as DQ requirements (Pipino et al., 2002), which represent generic concepts, should be identified in accordance with what customers need to know: for example, customers could state that their data should be accurate, or complete, or on time. On the other hand, low-level DQ requirements address the degree of accuracy required, or the required level of completion, or how long it could be delayed and still remain usable. To this end, the DQ characteristics that are observable on data (e.g. those appearing in ISO 25012 (ISO-25012, 2008)) must be specified. Commonly, DQ characteristics (what it should be observed in the data) can be mapped 1:1 to DQ dimensions (i.e. that customers need to measure). To effectively measure a DQ dimension on an item of data, for each of the DQ characteristics involved, certain measurable attributes have to be identified (cf. Definition 2.5.3).

Definition 2.5.3. Measurable attribute: a distinctive attribute of a group of one or various features related to a data quality characteristic that are involved in the measurement of data quality dimension.

Given that the definition of a quality requirement being met is related to the idea of zero defects, a customer must specify whether a data is defective or not by expressing some acceptance criteria that should be part of the low-level DQ requirement. For example, customers can decide that their data is defective with regard to completeness, when the ratio of incomplete records is lower than 90%. Thus, in order to assess the level of quality of a piece of data or of a dataset, the DQ experts should define explicit and customized measurement procedures for each DQ dimension that take into account the corresponding measurable attributes. These measures will later be used against the low-level DQ requirement to decide whether the data is defective or not.

Definition 2.5.4. Data Quality Management: is the set of activities aimed at ensuring that the data can really meet the requirements.

In the case when data could be considered defective, then some enhancement according to the DQ policies defined by the organization is required. This process first involves the identification of: the cause of the systematic production of defective data; the root causes (e.g. the collection process is failing, or the database was not appropriately designed); the specific feature of data which contributes towards the defect; and how the defect could be fixed. As a consequence, existing data requirements must be revised or even new requirements should be generated which specifically address the reparation of errors. This could be covered by the DQ management function of the organization.

Related to data quality management applied to business process, organizations need data to feed their business processes. As it is recognized, the successful of the instance of the process is grounded, among other factors, in the quality of the used data. Therefore, it is necessary to include data quality aspects in the description of the BP model in order

to systematically manage the level of quality of the data that flow through the process. Although certain data-quality related studies applied data quality management to BP, such as (Cappiello et al., 2013; Pipino et al., 2002; Rodríguez et al., 2012), they are only focused on the design of quality-aware BPs. This implies that there is a lack of proposals to give the necessary support for the *system configuration* and *process enactment* phases of the BP life-cycle.

2.5.2 High-level and Low-Level DQM Activities

As explained earlier, the high-level DQ management activities (control and/or assurance) are to use some low-level DQ management activities (measurement, assessment, decision-taking, enhancement). Further details are provided:

- **DQ Measurement:** the quantification of the level of data quality, following the principles of *meeting requirements*, needs the set of data requirements that data must meet. These data requirements are related with a set of DQ metrics that are relevant for the business. Both, the business expert and the DQ expert must reach an agreement on how to define the corresponding measurement methods for these metrics, and how to implement them. The implementation has to be parameterized in order to enable multiple measurements for various items of data of the BP.

A good way to increment the trustworthiness of data involve adding some extra information about the results of the measurement by an authoritative party. Specifically, and when the BP has to interchange data with an external entity (e.g. a provider), the certification could be carried out by an external, independent and authoritative entity in charge of guaranteeing a level of DQ, following the requirements described by ISO 8000-100 ((ISO, 2011a)).

- **DQ Assessment:** certain decisions concerning the convenience of using the data should be taken. The decision is to be made on the basis of the results obtained from DQ Measurement being compared to an objective threshold, stated by those business people who really know the processes, and the conditions in which the BP can run smoothly and seamlessly. These conditions commonly represent the context for the assessment of the data.
- **DQ Enhancement:** the data requirements necessary to transform, adapt, and/or change the data in order to better meet a specific level of DQ must be defined and communicated to the DQ Layer in order to let functionalities make the corresponding changes to the pieces of data so that requirements can be met. An example of enhancement can be the transformation of the format of some data, e.g. the date as “31/02/1985” (DD/MM/YYYY) instead of “02/31/1985” (MM/DD/YYYY). Further examples include: the identification of spelling mistakes in the names cities and countries or in dates, e.g. when the customer types “*Lndon*” instead of “*London*”. The improvements must be carried out based on the specific DQ dimensions under study, and on some patterns or sources that make these improvements, such as *Wordnet* (Princeton University, 2014), which is a lexical database of English that

can syntactically improve a word. Therefore, the set of requirements or patterns that this data must meet have to be defined.

Table 2.3 shows the relationship between these activities. How these high-level DQ management activities and low-level DQ management activities can be included in BP are analysed in Part V.

Table 2.3 – Actions depending on DQ Requirements

High-Level Activity	Low-Level Activity	Measure.	Assess.	Decision- Taking	Enhance.
Control		✓	✓	✓	×
Assurance		✓	✓	✓	✓

2.5.3 Data Quality Dimensions

In general, if somebody needs to control and/or assure the level of quality of a dataset, then some criteria, commonly named **data quality dimensions**, have to be chosen, and measurement methods have to be adapted for the specific context of the use of data and for each one of the chosen data quality dimensions as well as a method to aggregate all of the obtained values. Finally, an acceptance range must be defined for this last aggregation of the measures of the data quality dimensions. If the obtained value is inside of the acceptance range, then it is possible to state that the data set fits for the intended use.

Examples of data quality dimensions along with the definition have been proposed through the specialized literature, and it can be found in (English, 2001; Loshin, 2001; Pipino et al., 2002; Redman, 1998). Some of these data quality dimensions can be objectively measured, like completeness or accuracy, whereas other like believability or value-added require the subjective opinion of the user.

One of the most challenging tasks is the definition and adaptation of the measurement method for each data quality dimensions. Although some measurement methods have been already developed, the need for automating the process of measurement in a generalized way that can be adapted to different contexts of usage of data still lacks of a serious proposal. However, the measurement method would be more easily generalizable for those data quality dimensions which are objectively measurable (without the subjective opinion of the users of the data). In this case, the literature reinforces the idea of measuring the dimensions of completeness and accuracy against some data requirements (Benson and Hildebrand, 2012a). Designing a method accepting definition of data requirements and the data set, it will be possible to compute measures for completeness and for accuracy. Even more, this method could be externalized and executed by a third authoritative part, which, in addition, could even certify the level of quality of a data set. This would be really useful to increase the trustworthiness on the data being used for the task.

2.6 Case Study: Trip Planner

In order to understand the weakness of the imperative and declarative proposals, we introduce a case study where the objective of a BP is to obtain the data corresponding to the best product (outcome data of the process) in terms of the customer requirements. The difficulty is that each customer can have different requirements, and the same BP model needs to satisfy the necessities of each of these customers.

We can find in the literature several examples where data management in BP is crucial for the successful execution of a process, such as the on-line Book Store, which is based on a sale and delivery process (Goedertier and Vanthienen, 2006; Knuplesch et al., 2010; Rychkova et al., 2008a; Sadiq et al., 2007). However, the main scenario used in this thesis dissertation is the Travel Booking Example presented in (OMG, 2011a). This example highlights the lack of graphic representation and support for the combination of activities to search the concrete values for the data handling that achieve a common objective.

If we look through the Travel Booking Diagram (Chapter 9, page 28 in (OMG, 2011a)), the customer wants to book a flight and a hotel room. The process starts with (i) the travel reservation request; follows with (ii) the searching of flights, and hotel rooms, that compose the trip package that fits customer preferences; (iii) the trip package is offered to the customer; and finally, (iv) the customer, depending on his/her preferences, either formalizes the proposed trip package or cancels it (see Figure 2.4). The problem in this model lies in part (ii), where the search for each component of the trip is performed. This search is in accordance with the customer requirements: the place to visit, the possible dates, and the price, and it remains impossible to represent in an imperative way at design time.

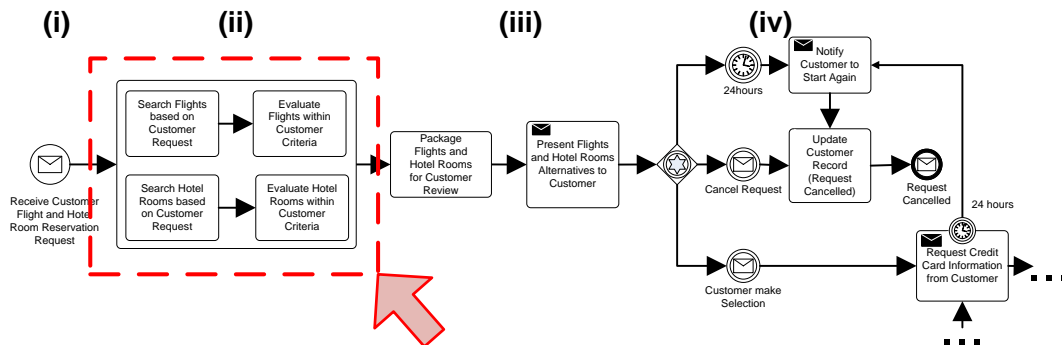


Figure 2.4 – Trip Planner Example (OMG, 2011a).

A parallel search is only possible when the customer request is formed by a concrete data input, in other words, when each data input has only one possible value (atomic value) and there is no relation between the customer criteria and the final result of both activities. It means that there are no relations between the activities, so they can be executed in a parallel way. For example, the customer wants to organize a trip, where (s)he wants to depart on “2014-09-15” and return on “2014-09-30”, with a flight from Seville to London, and wants to book a hotel room in London during these days. The dates and the locations given by the customer are the data input. These data input have

atomic values for the flight and hotel room searches, hence both searches can be performed in an independent way, by means of parallel way. However, if the customer is looking for a transoceanic flight, then the flight arrives the next day of departure. This implies that the check-in date in the hotel varies, depending on the output data of the flight (arrival date) in this case.

Generally, the customer searches by hand on the Internet for the cheapest combination of flight and hotel, for specific dates and cities. In addition, if the results obtained remain unsatisfactory, then other dates are sought until a convenient combination is found. Moreover, if necessary and also cheaper, a car can be rented in order to drive to another city to take a flight from another airport, thereby expanding the range of cities of departure and arrival. Taking a direct flight without renting a car does not necessarily mean that the trip is the cheapest option. For example, to go directly from Seville to London, a flight costs 250 euros, while a flight from Malaga to London costs 75 euros, and renting a car from Seville to Malaga costs 35 euros. As a result, the best option is to rent a car from Seville to Malaga and to take the flight from Malaga to London.

In order to obviate the search for several combinations of cities and dates, it is possible to combine the activities that represent the search of the trip package into a single BP, which we called the **Trip Planner**. The question becomes how to determine that the best trip is found (the business product obtained as data output), and which BP model obtain this optimization. Unfortunately, this cannot easily be solved in current business process models, since: (i) the input values of the data of the activities are inter-related, (ii) the output of every activity participates in the objective function to be minimized; and (iii) the optimization cannot be developed separately, since the decision of one activity influences the execution of the others.

Part III

Contribution I: Combi-BP Specification

Chapter 3

Specifying Data-Oriented Optimization in Business Processes

*First, solve the problem.
Then, write the code.*
John Johnson.

3.1 Context and Motivation

The cornerstone of BPM is the explicit representation of business processes with their activities, and the execution constraints between them. Generally, the explicit representation is described by imperative specifications, which described exactly how activities have to be performed. However, this explicit representation is not always possible to be described neither defined in an easy way at design time. The BP may be exposed to different environments and subjected to many conditions in which the execution order of activities cannot be described at design time in an explicit way. The importance of the BP models design resides in their influence over the remainder phases of BPM life-cycle, since the models are the bases of their later deployment and execution in a BPMS. Declarative specification appears as the best alternative for flexible specification when this execution order of activities cannot be described at design time. This flexibility comes from the specification of *what* has to be done instead of *how* has to be done.

This thesis dissertation is focused on environments where the BP model depends on the data handle in each instance. It means that the order of the activities and how they are executed depends on the data that the activities share, by means of inputs and outputs data. This dependency comes from the necessity to combine a set of activities and data, that flows through the process, in order to obtain a common goal. This goal can be the optimization of an objective function related to the outcome data, which enables to obtain the best business product. The main problem of represent this type of processes is that most current proposals of imperative and declarative languages fail

in two aspects: (1) they are only oriented towards the execution order of the activities, and remains unconcerned about how the data exchanged between the activities can affect the successful execution of the process; and (2) the optimization is oriented towards the minimization of the execution time or the resources of the business process, but not towards the optimization of the outcome data (business product) of each instance.

In order to solve the aforementioned limitations, we firstly establish a formalization to define the scope of our proposal, by means of the scope of data-oriented optimization problems in BP. Based on this formalization, two proposals are presented: (1) a declarative language with the aim of represents graphically the model which includes the process requirements referring to data description; and (2) an extension of the imperative language Business Process Model and Notation (BPMN). The aim of this extension is to enlarge the capacity of the standard in order to include the declarative specification of data-oriented optimization problems, into the imperative models described with the standard. The proposed language covers the user requirements (external information) with the activities and dependencies that business experts define in the declarative process (internal components). How the internal components can be configured to satisfy the external requirement is the Contribution II.

The chapter is organized as follows: Section 3.2 formalizes the data-oriented optimization problems dealt in this thesis dissertation. Section 3.3 introduces a proposed declarative language with data aspects. Section 3.4 defines the BPMN extension to permit the inclusion the declarative model formalized in the first section about data optimization in an imperative model. In order to illustrate the utilization of the various approaches, the case study related to a Trip Planner is formalized, graphically represented with DOODLE, and included in a BPMN model through CombA Sub-process in each corresponding section. Section 3.5 includes certain relevant related work. Finally, conclusions are drawn in Section 3.6.

3.2 Formalization of Data-Oriented Optimization in BPs

A formal definition is essential for clarification of the proposal, since it enables the identification, description, and definition of the type of problems to be studied. The main purpose of data-oriented optimization in BP is to obtain the combination of data to optimize the business product of an instance. As commented, there are models where the input data, output data, and the activities to be achieved for an objective are all known, but not how they should be combined to attain this objective. In order to introduce the formalization of the external requirements, and to present the activities that are combined, the components of the data-oriented optimization process (see Figure 3.1) are divided into two parts: (i) the internal components associated with the activities; and (ii) the external components that enables the communication with external processes. These external components describe the relation of the external requirements with the activities of the process through the data-flow (DF). The DF is the information that flows through the process and is formed by a set of variables. Since the variables of the data-flow can-

not be assigned directly to the activities involved, it is necessary to describe the internal components, and how they are connected to these external requirements:

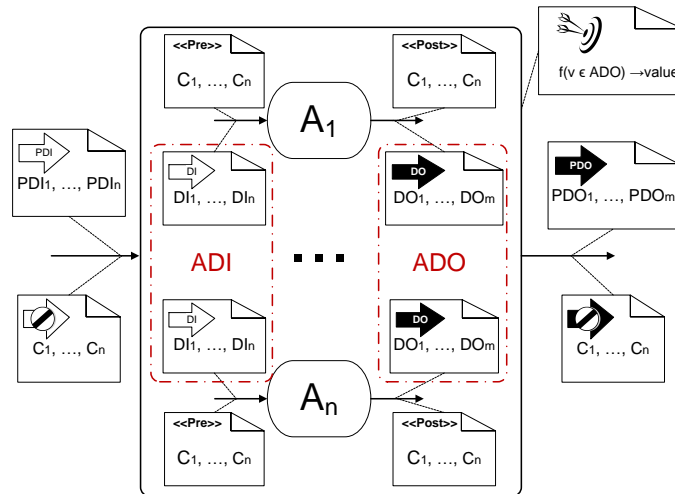


Figure 3.1 – External and Internal Components of the Data-Oriented Optimization Process

(i) The internal description of the components associated with the activities of the data-oriented optimization process includes:

– A , a finite set of activities $\{A_1, \dots, A_i, \dots, A_n\}$ contained in a BP. For each activity, we define:

* **ACTIVITIES_DATA_INPUT(ADI)**, all the input variables of the activities involved in the data-oriented optimization process, whereby $DATA_INPUT(A_i)$ represents the set of data input of an activity A_i .

$$\forall A_i, DATA_INPUT(A_i) \subseteq ADI, \text{ where } ADI \subset DF.$$

Since various activities can share some data input, then it is also possible that:

$$DATA_INPUT(A_i) \cap DATA_INPUT(A_j) \neq \emptyset, \text{ for } i \neq j$$

The union of all the $DATA_INPUT$, belonging to all the activities, constitutes the set ADI , with non-repetitive elements.

$$ADI = \{DATA_INPUT(A_1) \cup \dots \cup DATA_INPUT(A_n)\}$$

* **ACTIVITIES_DATA_OUTPUT(ADO)**, all the outputs of the activities that conform the process, whereby $DATA_OUTPUT(A_i)$ represents the data output of an activity A_i .

$$\forall A_i, DATA_OUTPUT(A_i) \subseteq ADO, \text{ where } ADO \subset DF$$

The union of all the $DATA_OUTPUT$, belonging to all the activities, constitutes the non-repetitive set ADO .

$$ADO = \{DATA_OUTPUT(A_1) \cup \dots \cup DATA_OUTPUT(A_n)\}$$

- * $\text{PRE}(A_i)$, the set of constraints that limits the specific values of the $\text{DATA_INPUT}(A_i)$ that satisfy the execution of activity A_i .
- * $\text{POST}(A_i)$, the set of constraints that limits the specific values of the $\text{DATA_OUTPUT}(A_i)$ that is satisfied after the execution of activity A_i .

(ii) The external description of the components of the data-oriented optimization process are:

- $\text{PROCESS_DATA_INPUT}(\text{PDI})$, the set of input variables of the declarative process, which determines the external requirements defined by the customer.

$$\text{PDI} \subset \text{DF}$$

- $\text{PROCESS_DATA_OUTPUT}(\text{PDO})$, the set of output variables of the process. PDO is formed of a subset of ADO that represents the business outcome of the process.

$$\text{PDO} \subseteq \text{ADO}$$

- INPUT_CONSTRAINTS represents the set of constraints that relates some PDI variable with some DATA_INPUT variable of the involved activities.
- $\text{OUTPUT_CONSTRAINTS}$ represents the set of constraints that relates some data (variable) of PDO with some data (variable) of DATA_OUTPUT of the involved activities.
- $\text{OBJECTIVE_FUNCTION}(\text{OBJ_FUNC})$ is an optimization function defined in terms of the data output of the activities (ADO) that satisfies the pre-conditions and post-conditions of the activities, and the input and output constraints. The objective of this optimization function can be to maximize or minimize the output data that represent the business product.

$$\text{OBJ_FUNC} : f(v \subseteq \text{ADO}) \rightarrow \text{value}, \text{ where } f \text{ is } \text{MAX} \text{ or } \text{MIN}$$

In order to solve the objective function, the $\text{DATA_INPUT}(A_i)$ values that optimize the output of the objective function must be found.

This formalization, used as a data-oriented optimization processes, includes the data of the process, the data of the activities, the objective function, and the constraints that enable the business expert to describe the possible data values and the dependencies of the activities. These constraints are defined by the following grammar, where **Variable** and **Constant**, can be defined in the Integer, Natural, Float, Dates, and String domains. On the other hand, **Set** can be defined as a set of **Constant** values of a specific **Variable**.

```

Constraint := Atomic_Constraint BOOL_OP Constraint
           | Atomic_Constraint
           | '¬' Constraint
           | Variable SET_FUNCTION Set
BOOL_OP:= '∨' | '∧' | '→'
SET_FUNCTION:= '∈' | '∉'
Atomic_Constraint:= function PREDICATE function
function:= Variable FUNCTION_SYMBOL function

```

```

| Variable
| Constant
PREDICATE:= '=' | '≠' | '<' | '≤' | '>' | '≥'
  {For the String domain only '=' and '≠' are allowed}
FUNCTION_SYMBOL:= '+' | '-' | '*' | '/'
  {These operators are only applicable to Numerical variables}

```

Specifically, in the data-oriented optimization process the variables of the constraints are the data defined by the external and internal components, i.e. *PDI*, *PDO*, *ADI*, and/or *ADO*, whose values will be established at runtime for each instance.

3.2.1 Formalization Applied to the Trip Planner

In this subsection, the formalization is applied to the Travel Search sub-process of Subsection 2.6 in order to form the cheapest trip package that fits customer preferences.

In the example, there are eight PDIs whose values are given by the customer: **departingFrom**, **setDepartingFrom** (the set of possible departure cities for the flight), **goingTo**, **setGoingTo** (the set of possible arrival cities for the flight), **earlyDepartDate** and **lastDepartDate** (the earliest and the last day when the customer prefers to depart respectively), and **earlyReturnDate** and **lastReturnDate** (the earliest and the last day when the customer prefers to return).

Given a data input, each activity calculates the price. The activities (A_i) and their ADI are detailed below.

- Flight Search Activity (A_F) returns the price of flights for a tuple of values for the data input.

$$\text{DATA_INPUT}(A_F) = \{\text{departingFrom}, \text{goingTo}, \text{departDate}, \text{returnDate}\}$$

$$\text{DATA_OUTPUT}(A_F) = \{\text{priceFlight}, \text{flightInformation}, \text{Data_Input}(A_F)\} \text{ where}$$

$$\text{flightInformation} = \{\text{outwardArrivalDate}, \text{returnArrivalDate}, \text{seat}, \dots\}$$

Certain existing pre- and post-conditions include:

$$\text{PRE}(A_F) = \{\text{departDate} \geq \text{systemDate} \wedge \text{returnDate} \geq \text{departDate}$$

$$\wedge \text{departingFrom} \neq \text{goingTo}\}$$

$$\text{POST}(A_F) =$$

$$\{\text{flightInformation} \neq \text{null} \rightarrow \text{priceFlight} > 0 \wedge \text{flightInformation} = \text{null} \rightarrow \text{priceFlight} = 0\}$$

- Hotel Search Activity (A_H) is employed to determine the cost of booking a hotel room.

$$\text{DATA_INPUT}(A_H) = \{\text{location}, \text{checkInDate}, \text{checkOutDate}\}$$

$$\text{DATA_OUTPUT}(A_H) = \{\text{priceHotel}, \text{hotelInformation}, \text{Data_Input}(A_H)\}$$

Certain existing pre- and post-conditions include:

$$\text{PRE}(A_H) = \{\text{checkInDate} \geq \text{systemDate} \wedge \text{checkInDate} < \text{checkOutDate}\}$$

$\text{POST}(A_H) =$
 $\{hotelInformation \neq null \rightarrow priceHotel > 0 \wedge hotelInformation = null \rightarrow priceHotel = 0\}$

- Car Rental Search Activities (A_{CR1} and A_{CR2}) are employed to determine the price of renting a car. Two cars can be rented during the trip, one at the source (A_{CR1}) and another at the destination (A_{CR2}). Nevertheless, the price of renting both cars is represented by A_{CRx} , where x can take the values 1 or 2, and depends on these entries:

$\text{DATA_INPUT}(A_{CRx}) = \{departingFrom, goingTo, departDate, returnDate\}$

$\text{DATA_OUTPUT}(A_{CRx}) = \{priceCarR_x, carR_xInformation, Data_Input(A_{CRx})\}$

Certain existing pre- and post-conditions include:

$\text{PRE}(A_{CRx}) = \{departDate \geq systemDate \wedge departDate < returnDate\}$

$\text{POST}(A_{CRx}) = \{carR_xInformation \neq null \rightarrow priceCarR_x > 0 \wedge carR_xInformation = null \rightarrow priceCarR_x = 0\}$

Therefore, the outputs of the process, PDO , are the outputs of the activities, which contain the information about the various components of the trip, as well as the total price of the trip ($price$).

There are several constraints that restrain data input and output that belong to the process and the activities. The constraints for the input and output data that determine the problem are defined below:

- **INPUT_CONSTRAINTS:**

- The constraints that define the possible values of departure date ($IC1$) and return date ($IC2$) of the flights, have to satisfy those described by the customer in the input data.

$(IC1) \text{ earlyDepartDate} \leq A_F.\text{departDate} \leq \text{lastDepartDate}$

$(IC2) \text{ earlyReturnDate} \leq A_F.\text{returnDate} \leq \text{lastReturnDate}$

- The constraints that describe the possible values of the departure airport ($IC3$) and arrival airport ($IC4$) of the flight, have to satisfy the possibilities of the input data proposed by the customer.

$(IC3) A_F.\text{departingFrom} \in \text{setDepartingFrom}$

$(IC4) A_F.\text{goingTo} \in \text{setGoingTo}$

- The date of check-in into the hotel has to coincide with the arrival date of the outward flight ($IC5$).

$(IC5) A_H.\text{checkInDate} = A_F.\text{outwardArrivalDate}$

- If the flight does not depart from the departure location ($IC6$), then the rental of a car (CR_1) is necessary.

$$(IC6) \text{ departingFrom} \neq A_F.\text{departingFrom} \rightarrow \{ A_{CR1}.\text{departingFrom} = \text{departingFrom} \wedge A_{CR1}.\text{goingTo} = A_F.\text{departingFrom} \wedge A_{CR1}.\text{departDate} = A_F.\text{departDate} \wedge A_{CR1}.\text{returnDate} = A_F.\text{returnArrivalDate} \}$$

- If the flight arrives at the destination city ((IC7)), then it is not necessary to rent a car at the destination city.

$$(IC7) \text{ goingTo} = A_F.\text{goingTo} \rightarrow A_F.\text{goingTo} = A_H.\text{location}$$

- OUTPUT_CONSTRAINTS:

- The total price is the sum of all the prices returned by the activities, as presented in constraint (OC1).

$$(OC1) \text{ totalPrice} = \text{priceFlight} + \text{priceHotel} + \text{priceCarR}_1 + \text{priceCarR}_2$$

In this example, the optimization involves the minimization of the cost of buying flight tickets, staying in a hotel room, and renting cars for the departure and arrival cities, and for the chosen departure and return dates.

$$\text{OBJ_FUNC} : (\text{MIN}, \text{totalPrice})$$

3.3 Data-Oriented Optimization Declarative Language (DOODLE)

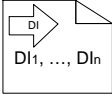
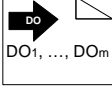
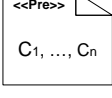
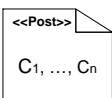
In order to facilitate the creation of models that follows this formalization into a BP model, we have defined a declarative language called DOODLE (Data-Oriented Optimization Declarative Language). The declarative description includes the data process, data activities, objective function and the constraints that let the business experts describe the possible data values in a declarative way. A graphical notation is also defined in order to include these components into a BP model easily. Although there is already a declarative language called DOODLE, presented by Cruz in (Cruz, 1992), it is a visual and declarative language for object-oriented databases, while our proposal is focused on a definition of a BP in a declarative way. All these components of the model are graphically represented by the symbols shown and detailed in Tables 3.1 and 3.2.

3.3.1 DOODLE Applied to the Trip Planner

Figure 3.2 shows how the components related to the Trip Planner Search sub-process, explained as the case study of this thesis dissertation, are represented with DOODLE.

Thanks to use DOODLE, the data-oriented optimization problems in BP can be specified easily. However, this kind of processes makes sense within a more complete process, where one of the main activities is the search of the optimal value. As shown in the case study, the data-oriented optimization problem corresponds to the sub-process of searching the trip package. Once the trip package is found, the process of booking the parts of the package, the payment, and even, the monitoring of the whole trip can be part of the

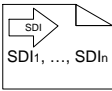


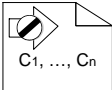
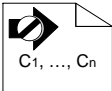
Table 3.1 – Internal Components associated to the activities of the declarative model

Symbol	Name	Description	Type
	Data Input (DI) of the activity	Set of data input of an activity	List of variables
	Data Output (DO) of the activity	Set of data output of an activity	List of variables
	Pre-condition	Set of constraints that represents the values of the DI that satisfy the execution of the activity	Constraints
	Post-condition	Set of constraints that represents the values of the DO that are satisfied after the execution of the activity	Constraints

complete process. These parts can be represented imperatively since the order of the activities are known at design time, and can be easily represented. Therefore, as soon as the data-oriented optimization process could be integrated in any of the existing imperative languages, the strength of our proposal would be greater.

For example, the resulting data-oriented optimization process could be added to the diagram presented in (OMG, 2011a) replacing the had-hoc sub-process of searching the flights, hotel rooms, and rental cars with this new Travel Search Sub-Process (see Figure 3.3). The following subsection details how to integrate the declarative description within an imperative specification.

Table 3.2 – External Components of the declarative model

Symbol	Name	Description	Type
	Process Data Input (PDI)	Data input of the declarative sub-process that describe the external requirement in each instance	List of variables
	Process Data Output (PDO)	Data output of the declarative sub-process	List of variables
	Objective function	An optimization function in terms of data	Minimize or Maximize an objective variable
	Input Constraints (IC)	Set of constraints that relates the SDI with the DI of each activity of the sub-process	Constraints
	Output Constraints (OC)	Set of constraints that relates the SDO with the DO of each activity of the sub-process	Constraints

3.4 BPMN Extension for Data-Oriented Optimization processes

The main standard used to model BPs is Business Process Model and Notation (BPMN), proposed by OMG (OMG, 2011b) and detailed in Section 2.2.2. BPMN 2.0 specification wants to stress the different stages in which the modelling process is composed: *description*, *analysis* and *execution*. The *description* stage concerns the visible elements and attributes used in high-level modelling, in other words, the closest stage to the human level. The *analysis* stage contains all of the description stage and others, and it is closer to a software engine level. Both stages are focused on visible elements and a minimal subset of supporting attributes/elements. On the other hand, the *execution* stage focuses on what is needed to execute process models.

To the best of our knowledge, there are no solutions with BPMN that enable to represent data-oriented optimization processes, such as the formalized before (cf. Section 3.2). Although BPMN represents business processes, by means of Sequence Flows and Data Flows, the objective function and the constraints that relate the data can only be added through annotations. Moreover, the annotations cannot be mapped to executed code in an automatic way. Fortunately, BPMN gives mechanisms to extend its functionality with new components. These mechanisms include and define the typography, linguistic conventions and style of BPMN, which are respecting in the definition of our proposal.

For this reason, the aim of this section is to extend the expressiveness of BPMN 2.0 with a new type of sub-process to include the declarative elements described in previous section. The proposal is focused on the support of this combination of activities, whose

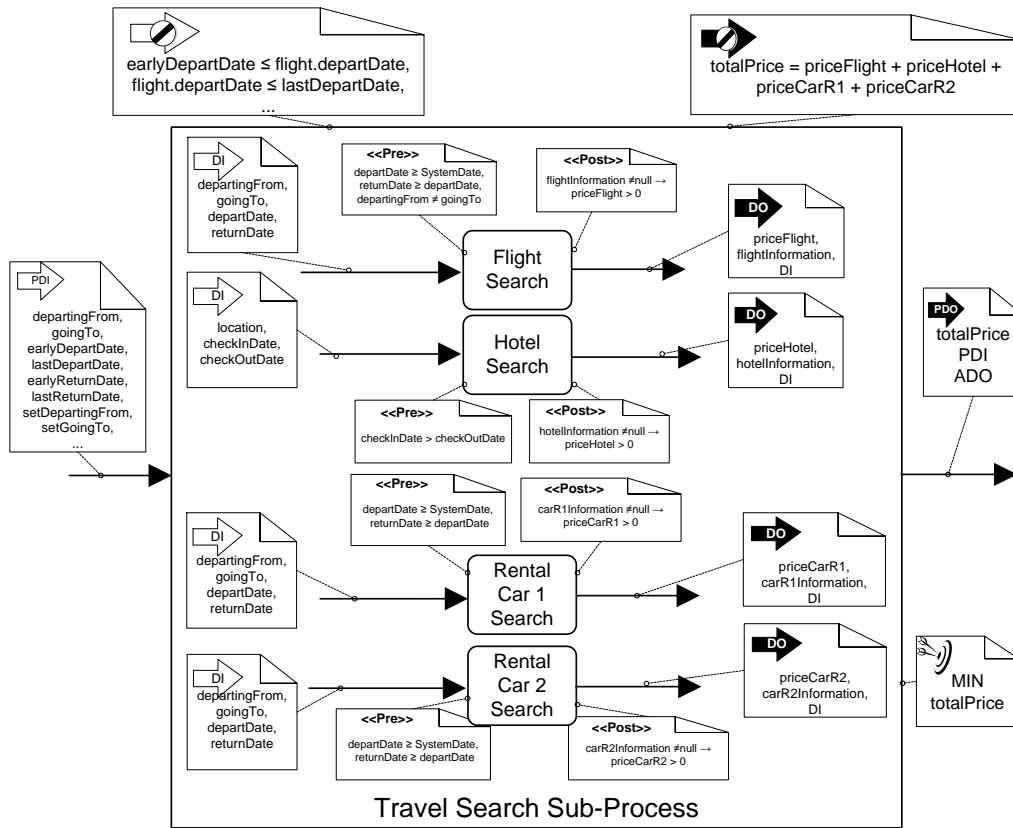


Figure 3.2 – Example of the trip planner described using DOODLE

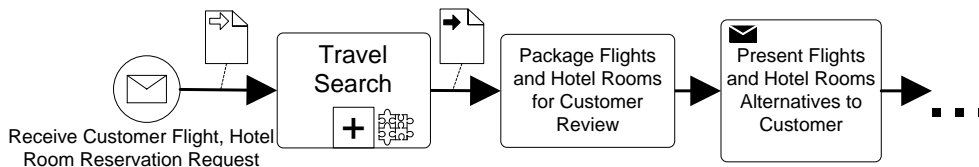


Figure 3.3 – Data-oriented optimization process included in the Trip Planner Process.

main purpose is to optimize the business product, and whose concrete values vary and depend on each instance. Thanks to this extension, declarative and imperative paradigms can be combined in a single model.

To do this, following subsections argues how the existing elements of BPMN cannot support data-oriented optimization processes, and thus, which solution is proposed. Specifically, a new metamodel is created in subsection 3.4.1, the new sub-process description is defined in subsection 3.4.2, the operational semantics are detailed in subsection 3.4.3, the event handling is analysed in subsection 3.4.4, and a solution to the *executable* stage is proposed in subsection 3.4.5. In addition, an editor capable of supporting our proposal is developed. Finally, this new extension is applied to the case study.

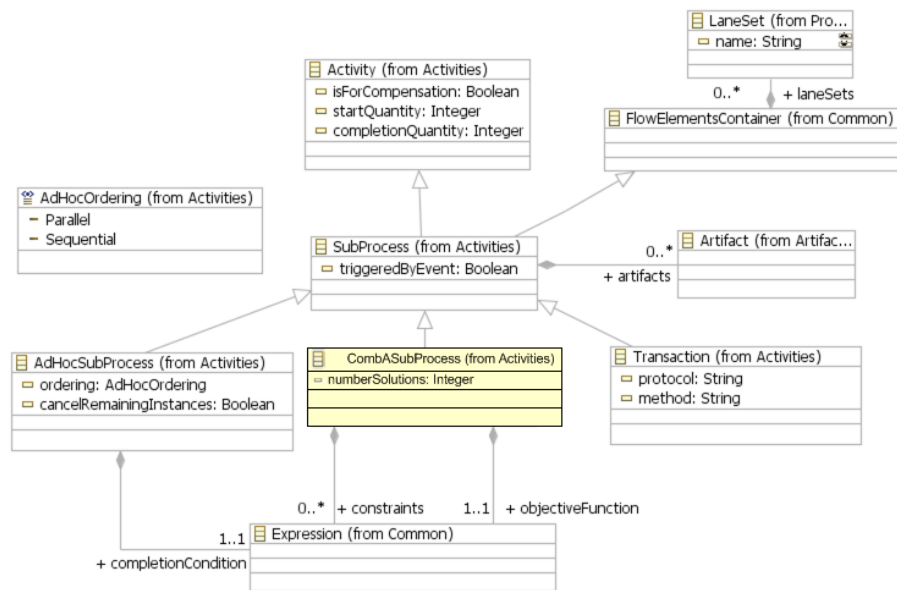


Figure 3.4 – Extension of BPMN 2.0 Sub-Process Metamodel.

3.4.1 Metamodel for the declarative sub-process

BPMN offers a way to represent the activities that have no REQUIRED sequence relationships through Ad-Hoc Sub-Process (OMG, 2011b). The Ad-Hoc Sub-Process semantics description is not enough for the combination of activities where the concrete values for the data have to be found to optimize the outcome data, and there is no way to determine them at design time. Although there is no explicit process structure in Ad-Hoc Sub-Process, some sequence and data dependencies can be added to the details of the process. In addition, the performers¹ determine when the activities will start, what the next activity will be, and so on. However, this is not valid when there are no sequence relationships, the data dependencies are on the data input and/or output of the activities and external requirements, and the performers cannot determine the sequence, as occur in data-oriented optimization processes. Furthermore, the list of BPMN elements that MUST NOT be used in an Ad-Hoc Sub-Process includes: Start and End Events, Conversations (graphically), Conversations Links (graphically) and Choreography Activities, which are useful to combine the activities to achieve an optimal goal.

In order to include these new requirements, an extension of Sub-Process definition (OMG, 2011b) is necessary. A new contextual scope is defined to achieve the optimization. Figure 3.4 shows the new metamodel related to Sub-Process, adding the new type of sub-process, that we called CombA Sub-Process², presented in this section.

According to the BPMN 2.0 methodology (OMG, 2011b), the *descriptive* stage is

¹A Performer defines the resource that will perform or will be responsible for an activity. The performer can be specified in the form of a specific individual, a group, an organization role or position, or an organization (OMG, 2011b).

²A different name is used to define the new sub-process with the aim of distinguish between the declarative language, called DOODLE, and the BPMN extension, called CombA Sub-Process.

necessary to define the characteristics of this new sub-process. This description enables the high-level modelling through a visible element and a set of attributes, as detailed in the following subsection.

3.4.2 CombA Sub-Process Definition as Declarative Component

CombA Sub-Process is a specialized type of Sub-Process which is a set of activities³ that have no REQUIRED sequence relationships. This new sub-process has to combine the activities in order to search the concrete values for the data handle that optimize a common objective. The set of activities can be defined in the process. However, the sequence and the number of performances for the activities cannot be determined by the performers of the activities, since the behaviour of an activity depends on the data belonging to other activities (or are external to the sub-process), and vice versa.

The formal definition of the declarative components presented in Section 3.2 is translated into BPMN elements in order to define the CombA Sub-Process. Therefore, the CombA Sub-Process is composed of:

- **Activities** (A in the formal definition): generally, these activities are tasks. A task is an unit of work, the job to be performed. However, each activity, in turn, can be another process. The unique requirement is that each activity has different and independent functionality.
- **Data Object**: it represents the information flowing through the process. The CombA Sub-Process handled mainly two types of data:
 - **Data Input**. There are two types of data input in the CombA Sub-Process: the first corresponds to the external input for the entire process (\mathbb{PDI} in the formal definition). It can be read by an activity and is given by a performer. And the second type corresponds with the data input of each activity that participate in the optimization (\mathbb{ADI} in the formal definition).
 - **Data Output**. There are two types of data output: the first one corresponds to the variable available as result of the entire process and the answer of the customer request (\mathbb{PDO} in the formal definition). And the second type corresponds to the output data of each activity (\mathbb{ADO} in the formal definition).
- **Constraints** (C in the formal definition): a set of Formal Expressions⁴ that relate both types of data object. The CombA Sub-Process handled mainly two types of constraints: (1) related to the process and activities (input and output constraints); and (2) related to each activities (pre and post-conditions).
- **Optimization Function**: The Formal Expression used to define the function to be optimized. This function relates the data output of the activities.

³An Activity is a Process step that can be atomic (Tasks) or decomposable (Sub-Processes) (OMG, 2011b).

⁴A Formal Expression is used to specify an executable Expression using a specified Expression language. A concrete constraint language is the one proposed in Section 3.2.

As the visible element for the modelling, we propose to use a puzzle piece symbol like the marker for the CombA Sub-Process and it can be used in a collapsed and expanded way (see Figure 3.5). The reason of choosing this symbol is that the activities in the CombA Sub-Process have to fit as the pieces of a puzzle, that can be a right visualization for the business level. The circular structure in the expanded CombA Sub-Process represents the lack of predefined order and sequence relationship, centred on the objective function, which is the CombA Sub-Process Core. The data and constraints are included as part of the definition of each activity and the sub-process itself.

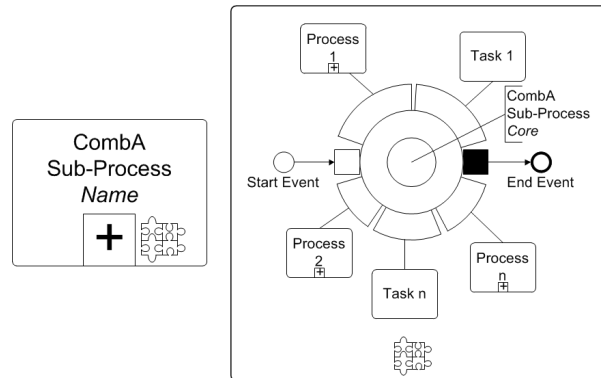


Figure 3.5 – Collapsed and Expanded representation of CombA Sub-Process.

The CombA Sub-Process element inherits the attributes and model associations of activities through its relationship to sub-process (OMG, 2011b), such as *properties*, and *sequence-flow*. Table 3.3 presents the **additional** attributes and model associations of the CombA Sub-Process.

Table 3.3 – CombA Sub-Process model attributes

Attribute Name	Description - Usage
data: set of Formal Expressions	Definition of the input and output data of the sub-process.
constraints: set of Formal Expressions	Definition of the input and output constraints that relate the data that flow through the Sub-Process, limiting their possible behaviours.
objectiveFunction: Formal Expression	Definition of the global optimization goal.
numberSolutions: Integer	Attribute to determine if the CombA Sub-Process searches only one of the best solutions or all the best solutions (values 0 or 1 respectively). The default value is 0.

In addition, the activity element should be extended to fit with the formalization. In order to differentiate this specialized activity, it is called CombA Task. Table 3.4 presents the **additional** attributes and model associations of this CombA Task.

Table 3.4 – CombA Task model attributes

Attribute Name	Description - Usage
data: set of Formal Expressions	Definition of the input and output data of the activity.
constraints: set of Formal Expressions	Definition of the pre- and post-conditions of the activity, limiting its possible behaviours.

3.4.3 CombA Sub-Process Operational Semantics

Following the definition for Sub-Processes presented in (OMG, 2011b), a CombA Sub-Process is an activity that encapsulates a Sub-Process that is in turn modelled by Activities, Gateways, Events and Sequence Flow. Moreover, the CombA Tasks involved in the optimization could be composed of Conversations, Choreographies and other Sub-Processes. The CombA Sub-Process is instantiated when it is reached by a Sequence Flow token⁵ through an unique Start Event. The CombA Sub-Process instance is completed when the CombA Tasks achieve the optimal goal and none of its CombA Tasks are still activated, in other words, when the End Event is reached by the token. Since various combination of values can produce the same result, the integer *numberSolutions* attribute establishes whether the CombA Tasks should find all the best solutions or only one. As long as the CombA Tasks do not find the optimal/s goal, and therefore, the concrete values for the data handle, the sub-process will not end.

3.4.4 CombA Sub-Process Handling Events

Thanks to incorporate the declarative model into a BPMN component, it is possible to use the handling events defined by the standard. For example, the events can help to control the spent time to find the optimal data solution. In order to control this kind of problems, BPMN provides a set of Timer Events. For example to represent that if after a period of time no optimal goal is obtained, then the operation is cancelled, returning the best solutions found until that moment. Sometimes, this timer requirement is provided by the customer who does not want to wait. However, timer requirement can also be used to provide faster products from the service provider point of view.

In general, BPMN provides several event handlers that help to manage and solve situations that happen during the course of a Process instance. Various of these event handlers can be applied to the CombA Sub-Process, especially intermediate events, since CombA Sub-Process can be affected by the same types of events than a common activity.

An Intermediate Event indicates where something happens (an Event), somewhere between the start and the end of a process. It will affect into the flow of the process, although this event will not start or (directly) terminate the process (OMG, 2011b). The

⁵The concept of a token is used by BPMN to facilitate the discussion of how Sequence Flows are used within a Process (OMG, 2011b). A token will traverse the Sequence Flows and pass through the elements in the Process. A token is a theoretical concept that is used as an aid to define the behaviour of a Process that is being performed. The behaviour of Process elements can be defined by describing how they interact with a token as it “traverses” the structure of the Process.

Intermediate Timer Event, specifically the interrupting type, interrupts the activity, to which is attached, changing the normal flow into an exception flow.

Applied to the CombA Sub-Process (see Figure 3.6), it implies that there will be two outcomes: successful completion and failed completion.

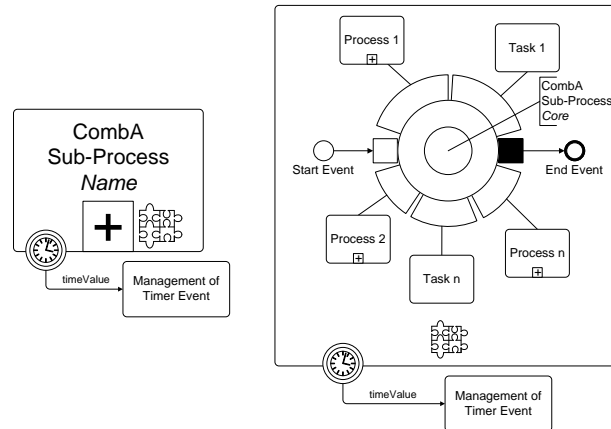


Figure 3.6 – A collapsed and expanded CombA Sub-Process with Timer Event.

Although in this thesis dissertation only Timer Events are detailed, a CombA Sub-Process can be combined with the set of Events given by BPMN 2.0 (e.g. Error, Message, Conditional, etc) (OMG, 2011b) and completed handling these possible events as it is done by sub-processes.

3.4.5 CombA Sub-Process Execution Semantics

BPMN 2.0 pays special attention to the execution semantics since there is an important need of executable process models. A CombA Sub-Process contains all the necessary elements to define a data-oriented optimization problem within a BPMN model. However, these components are defined in a declarative way, which implies certain treatment to transform them into executable. Specifically, CombA Sub-Process contains a number of embedded inner activities and is intended to be executed with more flexible ordering, compared to the routing of Processes or Ad-Hoc Sub-Processes. There are several ways to combine the activities and all of them depend on the performer (designer) criteria.

A complete study on how this CombA Sub-Process can become executable is provided in Part IV.

3.4.6 BPMN Editor including CombA Sub-Process

In order to provide a tool with the capability of defining CombA Sub-Processes within a BPMN model, CombiS-BP⁶ Editor is developed.

CombiS-BP Editor enables combined modelling of the two aforementioned specifications: (i) allows an imperative specification when experts know the execution order of the activities in the model, by means of the BPMN 2.0 components and; (ii) enables a declarative specification of data-oriented optimization processes through the declarative component: CombA Sub-Process. Therefore, CombiS-BP Editor is proposed with the aim of integrating imperative and declarative languages.

To the best of our knowledge, there are no available tools that enable the modelling of a BP where the order relation among the activities depends on the data values of each instance. In the search of existing tools that model declarative constraints in BP and workflows, various relevant tools were found ((Goedertier and Vanthienen, 2007; Kopp et al., 2012; Pesic et al., 2009; van der Aalst, 2004)). All of them use the value of the data to determine which task is executed next, deal with multiple instances, and extend the standard functionality. Although these tools define a set of constraints that relates the activities, the aim of these constraints is to establish an execution order, that in our approach is unknown until the value of the data are analysed at runtime.

Heretofore, with the existing BPMN 2.0 modelling tools, business experts had to decide the exact order (sequential, parallel, etc.) to model the part where the activities require combination and they could not know how to combine them. CombiS-BP Editor combines the imperative part of the model where experts know *how* things should be done, with the declarative part of the model where experts only know the activities that need to be executed, the relation between activities, and the optimization function.

In order to include CombA Sub-Process and to describe the declarative part of the process, the grammar used by CombiS-BP Editor is shown in Figure 3.7. CombA Sub-Process contains an *Identifier*, its functionality is described through a *Description* and its goal to be optimized is represented as *ObjFunc*. The set of activities to be combined are represented as *Activities*. Each *ActivityElem* from *Activities* represents the features of each activity (e.g. name, and attributes). Finally, the relationships between the activities are represented as *Constraints*, where some *DeclarationConstraints* are defined through *Data* (belonged to Process or to Activities).

CombiS-BP Editor has been developed as an extension of OPBUS tool ((OPB, 2012; Varela-Vaca et al., 2011)), which is an eclipse plug-in. CombiS-BP Editor integrates a BPMN 2.0 modeller that enables the creation of CombA Sub-Process (see Figure 3.8). The user interface is composed of four main parts: edition zone, palette, properties and problem tabs, and a project workspace zone with basic menus. Business experts model the common BP in the edition zone. In this part, declarative and imperative specifications are differentiated through the different elements used in the process. The palette provides the graphical definition of BPMN elements (imperative specification) and CombA Sub-process(declarative specification), which can be selected and dropped into the edition

⁶Since this editor covers the first step of Combi-BP framework, its name is associated to the phase of Combi-BP Specification (CombiS-BP)

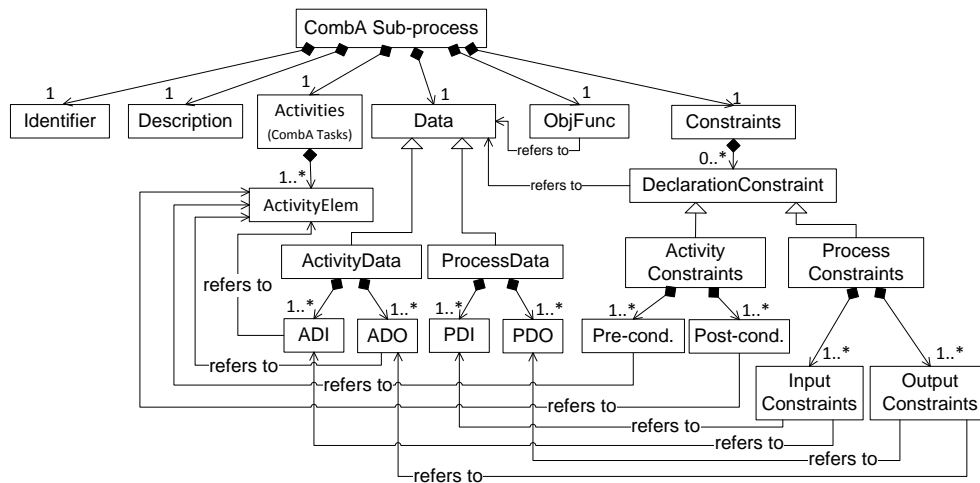


Figure 3.7 – Vocabulary in CombiS-BP Editor.

zone.

The marker associated to ComBA Sub-Process graphical definition in the palette is a set of puzzle pieces symbol (such as defined in 3.4.2), and to ComBA Task (activities involved in the combination), the graphical definition is a unique puzzle piece symbol. ComBA Flow is a solid line that can connect only ComBA Task with ComBA Sub-Process. The declarative definition is completed with the properties part, which provides support for the definition of elements details. In order to fill the ComBA Sub-Process grammar shown in Figure 3.7, some properties and elements are added. Specifically, the property *ActivityData* and *ProcessData* are added, to ComBA Task and ComBA elements respectively, through a *Data* element from the palette. In the same way, the set of *Constraints* are added to ComBA element and specify by the properties provided in the Constraint element (see Figure 3.8). Finally, the properties, provided in the properties part, associated to ComBA Sub-Process element are *Id*, *Description* and *Objective function*.

3.4.7 BPMN Extension Applied to the Trip Planner

The formal definition given in Section 3.2 for the Trip Planner Example is also valid for the ComBA Sub-Process definition. Therefore, the attributes of ComBA Sub-Process for the example are:

- **data:** All the input and output data of the process are part of this attribute.
- **constraints:** All the constraints defined as input and output constraints are part of this attribute.
- **objectiveFunction:** To get the cheapest trip, the objective function is to minimize the total prices, as defined in the formalization `OBJ_FUNC (MIN, totalPrice)`.
- **numberSolutions:** All the best solutions regarding the objective function to optimize. According to the values given in Table 3.3, the value could be 1.

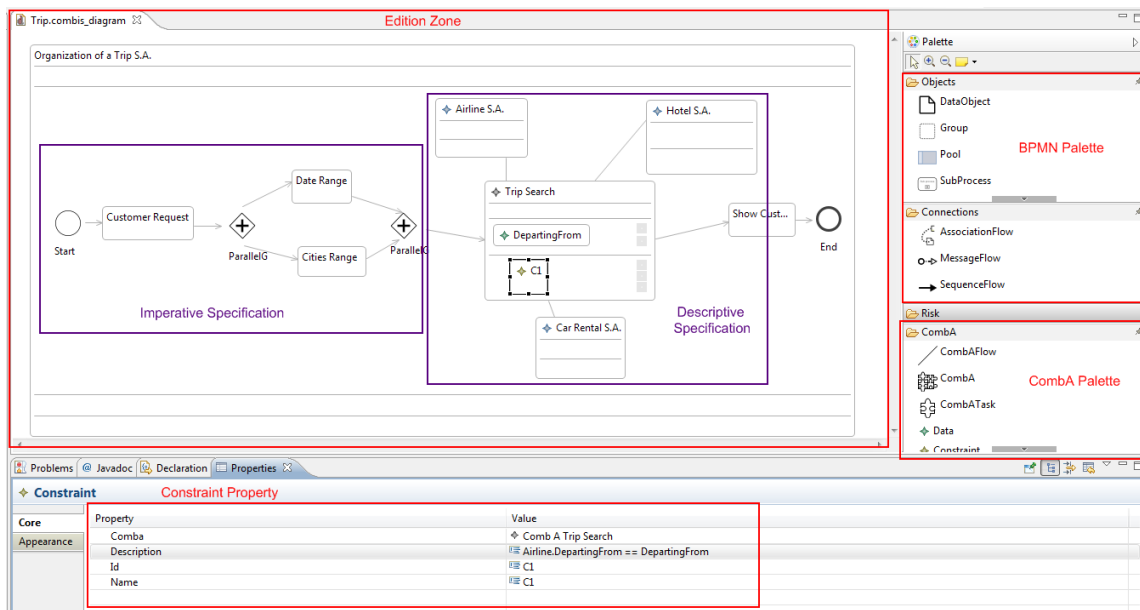


Figure 3.8 – CombiS-BP Editor.

In addition, the activities Flight, Hotel and Car Rentals Search Activities, are specialized in Comba Tasks, whose new attributes correspond to:

- **data:** All the input and output data of each activity are part of this attribute.
- **constraints:** All the pre- and post-conditions of each activity are part of this attribute.

Finally, Figure 3.9 depicts the way to represent the combination of these activities with the new Comba Sub-Process marker.

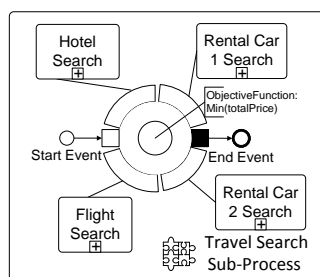


Figure 3.9 – Expanded Search Travel Comba Sub-Process.

3.5 Related work

In order to analyse the related works applied in this contribution, it is necessary to study how the existing imperative and declarative languages support and consider the data-oriented optimization processes as formalized before.

Related to the imperative specification, Yunzhou Wu et al. in (Wu and Doshi, 2008) show how the constraints that necessitate coordination may be represented in the Business Process Execution Language for Web Services (BPEL). BPEL is an OASIS standard executable language for specifying actions within business processes with web services (OASIS, 2005). Processes in BPEL export and import information by using web service interfaces exclusively. But there is not a protocol defined in BPEL to combine the activities nor to select the concrete values of the data input according to the optimization function. The authors in (Wu and Doshi, 2008) use a generalized adaptation and constraint enforcement models to transform the traditional BPEL process into an adaptive process. However, the authors solve the combined adaptation and constraint enforcement models in order by obtaining a policy that recommends adaptive actions while respecting the constraints. Therefore, there is no combination to search the concrete values for the data handle in terms defined in our work.

Aside from this, service-oriented systems have emerged as the paradigm to provide such automated support for BP. Van der Aalst et al. in (van der Aalst et al., 2003) and Papazoglou et al. in (Papazoglou and van den Heuvel, 2007), present Web Services as the infrastructure to foster BP by composing individual Web Services to represent complex processes. There are several studies on the composition of services that can be extrapolated to the composition of activities in BPs since services are specific activities in the BP. To the best of our knowledge, none of these studies represents graphically or solves the type of coordination that this contribution presents: a combination of independent activities to find the concrete values of their data that optimize an overall objective.

On the other hand, various declarative languages can be found in the literature, most of which describe the correct order of the activities of the BPs. Compared with declarative languages analysed in Section 2.3 and shown in Table 3.5, our proposal DOODLE includes data-oriented optimization aspects in a declarative manner. It is done by means of a set of constraints that describe the data exchanged among the activities, when their relations cannot be defined explicitly at design time. The model and the reasoning framework use Constraint Programming in order to configure an imperative model and infer the possible values of the data and achieve the optimal outcome at run-time. However, every formalism for reasoning could completely valid whether enable efficient and complete reasoning algorithms to cover the process necessities. Related to the imperative and declarative description capacities, only few of them enables to combine and integrate both types of description in the same model. This characteristic enables a more flexible and adaptable specification, letting the business experts to describe parts where they know how to model them in an imperative specification, and the parts where they know the BP requirements in a declarative way. One of the mains strength of DOODLE is its capacity to construct an imperative model and also assist to the customer to decide which is the best activity to execute at runtime, since generally, the languages are oriented to only a use.

In other respects, the data perspective is more widespread. Although all these declarative languages include some information about data, none of them include the data input and output of the activities with the aim to optimize the object obtained from the BP being not the same for the specification of the pre- and post-conditions. This characteristic enables to define the requirements for the execution of the activities, which play an important role in the optimization process. Only Condec-R includes an objective function in its language. However, this objective function is oriented towards the optimization of resources and execution time, not to the business product. All these characteristics make DOODLE as a complete alternative to specify declarative requirements within imperative descriptions, and more specifically, to define data-oriented optimization problems in BP.

Table 3.5 – Declarative Languages Comparative including DOODLE

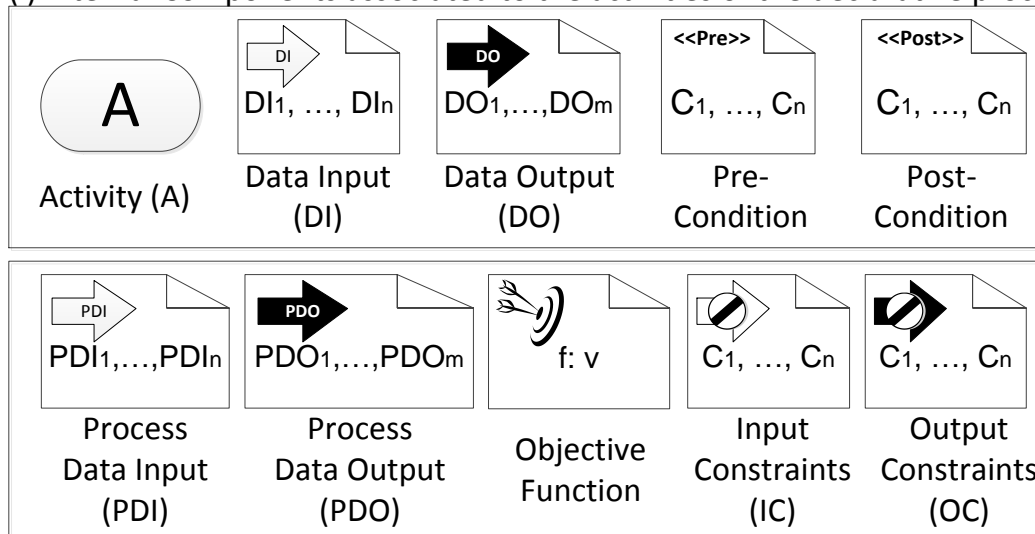
Language	Formalism	Imper. and Decl.	Use of model	Data persp.	Pre and Post	Opt. Funct.
Pocket of Flex.	Graph theory + algorithm	✓	Val.			
DeCo	First Order Logic	✓	Val.	✓	✓	
CRGs	Pattern matching		Val.	✓		
Em-Bra ² Ce	Color Petri Net	✓	Val.	✓		
Penelope	Event calculus	✓	Constr.			
ConDec	Linear Temporal Logic		Val.			
ConDec-R	Constraint Programming		Assist.	✓		✓
Data-aware	Event Calculus		Val.	✓		
DOODLE	Constraint Programming	✓	Constr. Assist.	✓	✓	✓

3.6 Summary and Discussion

In this chapter, the specification of data-oriented optimization problems in current BPM has been tackled. The main obstacles are related to the lack of mechanisms to define and specify data aspects oriented to the optimization of the business product. In the majority of cases, the role of data is mostly limited to describing the execution of not of an activity, which depends on the value of a variable of the data-flow. The proposals fail to take into account how the exchange of data between the activities can affect the successful execution of the process. We have analysed some of the most relevant declarative and imperative languages in business processes. From this analysis we have detected that none of them permit the declarative description of data in the business processes, and how it can influence in the obtained model.

In order to face this need, we have firstly formalized the data-oriented optimization processes. A declarative language, called DOODLE, is proposed. DOODLE permits the description of the data exchanged among the activities of the process in a declarative way by means of data, constraints, and an objective function, as shown in Figure 3.10.

(i) Internal Components associated to the activities of the declarative process



(ii) External Components associated to the declarative process

Figure 3.10 – DOODLE Graphical Components.

In addition, an extension of BPMN 2.0 has been proposed, to incorporate the declarative model into an imperative model. A Comba Sub-Process with its associated marker enables to incorporate combination requirements into a standard. These combination requirements will increase the scope of the expressive ability of business description. Finally, the proposal has been supported with the implementation of a plug-in that enables the graphical specification of the data-oriented optimization process.

Part IV

Contribution II: Combi-BP Transformation

Chapter 4

Configuration of an Imperative Business Process to minimize the execution time according to Data Dependencies

Before software can be reusable it first has to be usable.

Ralph Johnson

4.1 Context and Motivation

As commented in previous chapters, declarative models are commonly used to give flexibility in the description of business models. Unfortunately, the data dependency between the activities and how this can affect the correct execution have been overlooked in these declarative specifications. The declarative languages found in the literature pay attention to the activities order, for example activity *A* has to be executed after activity *B* is executed, or if activity *C* is executed, activity *D* cannot be executed eventually. However, we consider that sometime the activities order is determined by the data dependency requirements. It means that an activity is executed after another because it needs the data provided by the first one.

Although this dependency can be defined in a declarative way, as presented in Chapter 3, it could also be described in an imperative way. The problem is that the model can include data optimization aspects that are not covered by the imperative models. For this reason, we propose to deal with the data dependencies in two steps: (1) Activities configuration to minimize the execution time, and (2) Data-oriented model to optimize the process outcome. Combining both strategies it is possible to obtain an optimized outcome in a minimized time.

In order to clarify or proposal, let us use the simple example shown in Figure 4.1.

This example represents, in a declarative way, four activities whose input and output data are related by means of constraints. In this example, two problems arise: (1) which is the order to execute the activities, and (2) how to determine the values for the data handle to optimize the objective function. The first problem is found when for example, the designer has to decide which activity has to be executed before *A* or after *C*. The second problem arises when it is crucial to determine the values of I_1 and I_2 , since an input constraint establishes that $I = I_1 + I_2$ to maximize $O_1 + O_2$. In this chapter the first problem is faced, while the second problem is analysed in the next chapter.

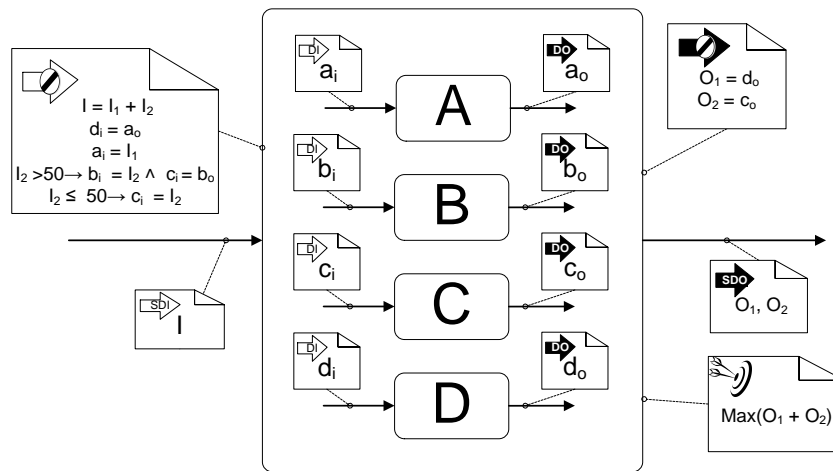


Figure 4.1 – Imperative BP Representation for Data-Oriented Optimization BP.

Since the data dependencies are defined in the declarative models, and they cannot change, we propose to build an equivalent model. The reason why we suggest the creation of an imperative are analysed in the Chapter 1. Principally, an imperative model is: (1) more understandable, (2) closer to executable models, (3) supported by several commercial BPMS, where they can be executed.

Since the declarative models are more flexible than the imperative models, it implies that several imperative models can compliance the same declarative model. This generates a hesitation about which is the more appropriate imperative model. We deal with this issue as a configuration problem, where the minimization of the execution time of the model. A configuration problem is an ordered arrangement of a set of parts, whose solution is the selection and arrangement of a set of parts that satisfy the problem requirements as well as constraints associated with these objects (Petrie, 2012). From a BP point of view, in a configuration problem, it is necessary to find the order relation between the activities and the control flows that related them.

For the example of Figure 4.1, an imperative model that compliance the data dependencies between the activities minimizing the execution time is shown in Figure 4.2.

Therefore, Figure 4.3 shows our proposal, where an imperative model equivalent to a declarative model is depicted. In order to model the BP in an imperative way, BPMN

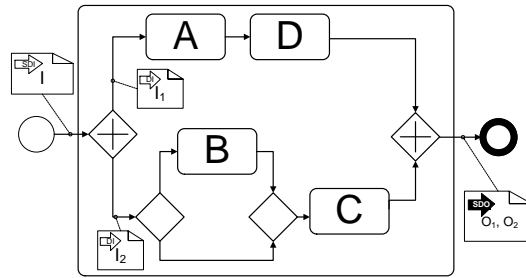


Figure 4.2 – Imperative BP Representation for Data-Oriented Optimization BP.

(OMG, 2011b) is chosen, since it is supported by several commercial BPMS. The task "Supply Input Data Values" provides the different combinations of input data values for the activities to obtain the optimal business product. The details of this activity are explained in Chapter 5. On the other hand, the sub-process "Execute sub-process" represents the imperative model that is configured according to data dependencies. This sub-process is formed of the set of activities involved in the declarative model by means of the BPMN connections and gateways. Several possible imperative models exist that satisfy the data dependencies, but our objective is to find the optimal imperative BP model with respect to execution time.

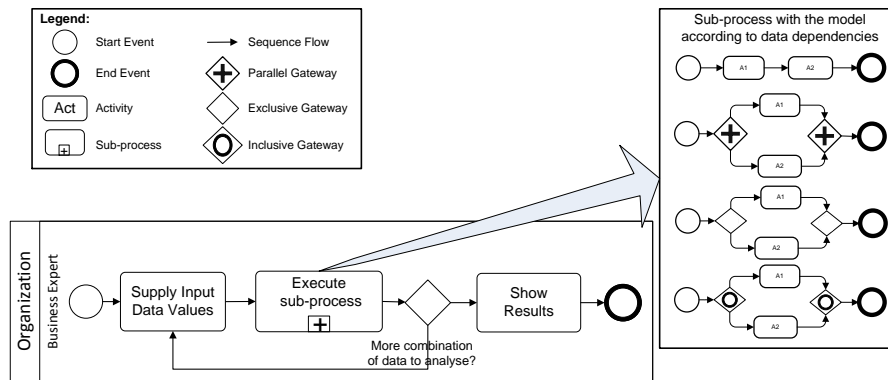


Figure 4.3 – Imperative BP Representation for Data-Oriented Optimization BP.

To the best of our knowledge, no solution has yet been found that based on a declarative specification focused on data management, enables an imperative BP to be derived from this specification to optimize the outcome data by minimizing the execution time. In this chapter, how to obtain this imperative model is analysed.

The rest of the chapter is organized as follows: Section 4.2 presents the configuration system, which also proposes a flexible and adaptable imperative modelling method. Section 4.3 details the automatic configuration from declarative to imperative modelling. Section 4.4 applies the proposal to the example. Section 5.4 describes related work. Finally, conclusions are drawn and future work is proposed in Section 4.6.

4.2 Configuration System Description

The main aim of the system explained in this chapter is to configure an imperative model from the declarative specification by establishing an execution order between the activities. This order is obtained by analysing the relationships between the data provided and used by the activities.

In order to describe a solution independent of any specific technology that can be used, the configuration system has been built inspired by the foundations of the Model-Driven Development (MDD) paradigm. The proposal is focused on a particular view: the OMG's Model-Driven Architecture (MDA) (OMG, 2003), which is a software design approach for the development of software systems. The configuration system therefore consists of various modelling stages, each with a separate abstraction level. We isolate the specification, which is independent of the platform, and then isolate the model, which is independent of the technology but not of the modelling language. The imperative language BPMN is chosen since it can be enacted in any of the existing BPMs on the market. And finally, we isolate the implementation, which is specific to a particular technology. Among other things, the same BP declarative specification can lead to various BP imperative models. Furthermore, the configuration system contains various levels of abstraction which make it easier to detect errors and omissions, to reason and validate, and to communicate with the many stakeholders in each level.

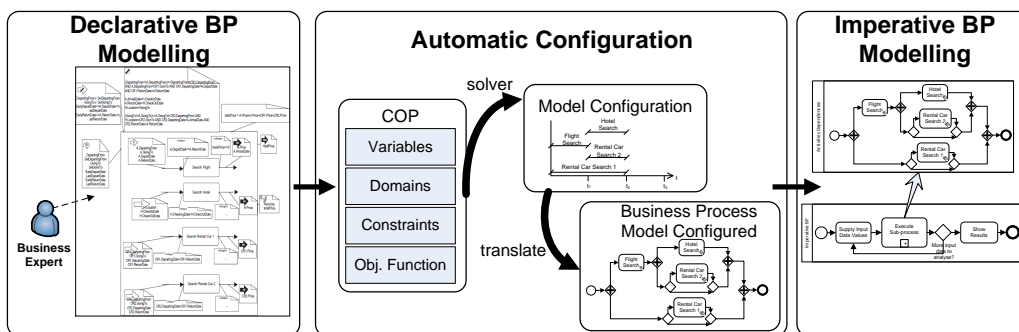


Figure 4.4 – Configuration System Architecture.

As shown in Figure 4.4, the Declarative BP Modelling represents this highest level of abstraction, since, through the elements specified in this stage, it must be possible to model all concepts that are available in all of the BP platforms supported (Imperative BP Modelling). The problem is specified in a declarative way by a set of business experts, who process the entire knowledge of the providers involved in the combination and are well informed about the BP requirement. The second stage consists of an Configuration System in charge of the transformation of this declarative specification into a specific model for a particular standard imperative languages for BP modelling. This transformation consists of defining a Constraint Optimization Problem which obtains the model configuration through minimization of the execution time of the process. The configuration determines the order in which the activities should be executed in the BP. Finally, Imperative BP

Modelling completes the third stage together with the functionality explained in chapter 5.

4.2.1 Relation between Data Dependencies and Imperative Models

The imperative model created by the Configuration System has to follow two main rules to compliance the declarative model with an optimal execution time: (i) Compliance the data dependencies (data provided and consumed by the activities), by means of satisfying the constraints, and (ii) Minimize the execution time of the model for a theoretical constant execution time for the activities. It means that there is no another imperative model that compliance the data dependencies in a less execution time.

Applied to the trip planner example, for when solely the activities “*Flight Search*” and “*Car Rental 1 Search*” are considered for the modelling configuration, the possibilities are shown in Figure 4.5.

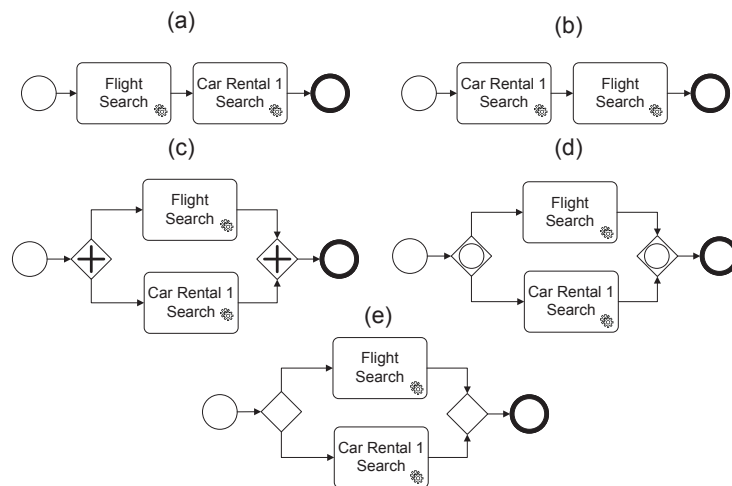


Figure 4.5 – “*Flight Search*” and “*Car Rental 1 Search*” Model Possibilities

A sequence model implies that an activity use the data provided by another activity, represented by Figure 4.5(a) and Figure 4.5(b). A parallel model, Figure 4.5(c), could be even better according to execution time, since these two activities could be executed at the same time. On the other hand, an inclusive gateway, Figure 4.5(d), implies that at least one activity is executed and an exclusive gateway, Figure 4.5(e), implies that only one activity is executed. For this example, the most optimal option is to execute the activities in a parallel way since they have no data dependencies, and there is no another imperative model that can execute both activities in less time.

This modelling combination and analysis increases considerably as soon as the number of activities grows. Hitherto, this configuration has been made by the business experts, we now propose passing this responsibility to the Configuration System. Therefore, in order

to transform the declarative model into an imperative model with a minimal execution time, we propose the use of the Constraint Programming paradigm, as explained in Section 4.3 below.

4.3 Automatic Configuration from Declarative to Imperative Model

The configuration of the imperative model from the declarative description is a difficult and hard task, since the flexibility and adaptability that only a declarative description can offer must be maintained. In our case, the configured imperative model has to be able to combine the activities with the aim of testing every possible combination of data, while taking into account the data dependencies between the activities and the objective function. In order to perform the optimal configuration, we propose the three steps shown in Figure 4.6.

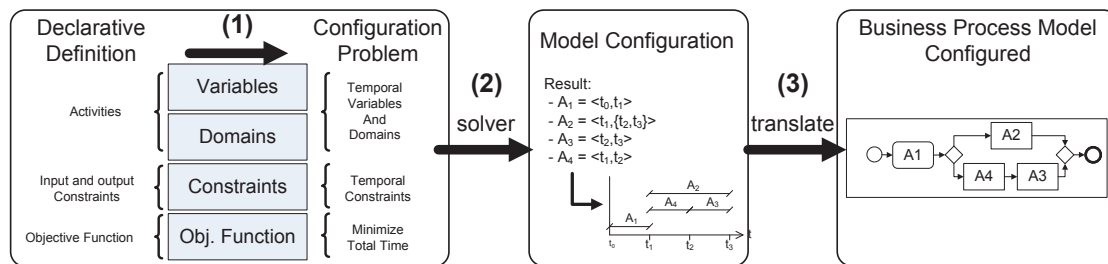


Figure 4.6 – Configuration from Declarative BP Model to Imperative BP Model

1. **Create a Constraint Optimization Problem from the declarative Model:** As explained in detail in Section 4.3.1, we propose transforming the declarative model into a Constraint Optimization Problem (COP) (cf. Chapter 2.4). A COP is another declarative model but can be computationally solved in an automatic way. It is necessary to highlight that a COP specification is very similar to the process models proposed here, since both are declarative models which define the problem but do not solve it.
2. **Solve the COP:** In order to solve the COP created above, a combination of search and consistency techniques is commonly used (Dechter, 2003). The consistency techniques remove inconsistent values from the domains of the variables during or before the search. Several local consistency and optimization techniques have been proposed as ways of improving the efficiency of search algorithms. This configuration problem is solved by using any of the existing CSP solvers; JsolverTM (Manual, 2003) in our case.

3. **Create the BPMN Model:** As explained in detail in Section 4.3.2, by using the results obtained from the COP, the imperative model can be created. The translation from the configuration results to a BP model implies to study the type of gateways that relates the activities, since, although two activities A and B could start at the same instant of time, perhaps there is a constraint which states that only one can be executed, not both. In that case, there can be an exclusive or inclusive gateway relating the two activities, and the decision between these two gateways is based on the study of the domains of the data related in the constraint: on whether there are overlaps or not.

4.3.1 Creating a COP from the declarative model

The relation between the input and output data of the various activities determines their relational order. As was commented, for the same declarative model, it is possible to find several configurations of activities that satisfy the requirements and/or the declarative description. We propose the use of an automatic transformation based on Constraint Programming to find the optimal BP model that maximizes the parallelization of the activities, with the aim of minimizing the execution time. In order to determine this configuration, we suggest the transformation of the declarative model into a COP, to analyse the possible instants when the activities can start and end their executions according to the data dependencies. In the declarative model, the numerical constraints are described to establish the dependencies between various activities. These dependencies represent the relationship between the input and output data which generally, imply and prompt a temporal order between these activities.

In order to model a possible execution order between the activities to find the optimal according to execution time, the possible value where an activity starts and ends is represented as the tuple $\langle t_{ini}, t_{end} \rangle$, where $t_{ini} < t_{end}$. Although the execution time of each activity is unknown at design time, the input and output dependencies remain significant and can be represented by the t_{ini} and t_{end} relation between the activities. The execution time of a BP depends on the execution time of each activity, and when they are executed depending on the control-flow structure. Since in our model, no information about the execution time of the activities is included, we pre-establish that each activity lasts the same interval of time; a unit of time t . Therefore, $t_{end} - t_{ini} \geq 1$, being possible values greater or equal to 1 in order to design more flexible models as explained below.

Related to the activity execution and how the imperative model can influence in it, in the worst case scenario, all the activities are executed sequentially, and the total execution time is the number of activities multiplied by the unit of time, $t * numberOfActivities$. In the best case, every activities can be executed in parallel, in which case the total time is t . Unfortunately, this best case is not always possible due to the data dependency between the activities.

In order to find the imperative model, whose instances are the less time consuming, the input and output data relationships of the activities of the declarative description are translated into a COP, by means of temporal constraints. For example, if there is a constraint that relates any input data of activity A with some out-

put data of activity B , then this means in the COP that activity A cannot start until activity B has finished. For example, in the trip planner example, the constraint $\{A_H.checkIn == A_F.outwardArrivalDate\}$ is translated into $t_{ini_{A_H}} \geq t_{end_{A_F}}$.

The idea of the transformation is based on that all the activities could start at the same instant unless there exists a relationship requiring one activity to start after another. Therefore, the optimization function of the COP would be the minimization of the total time, where the total time is the t_{end} of the last activity executed. Table 4.1 shows the relation between the declarative components of data-oriented optimization processes and the created COP.

Declarative	COP
Activity	$\langle t_{ini}, t_{end} \rangle$
Constraints (Input and Output Constraints)	Temporal Constraints
-	Minimize Total Time

Table 4.1 – Declarative model and COP elements relationship

As mentioned earlier, since there is no information about the duration of the activities, we have modelled it as a unit of time. The following question is the determination of the maximum execution time of each activity, with the same execution time for the complete model, with the aim of obtaining a more flexible and robust model. In order to study this problem, the example of Figure 4.7 is analysed.

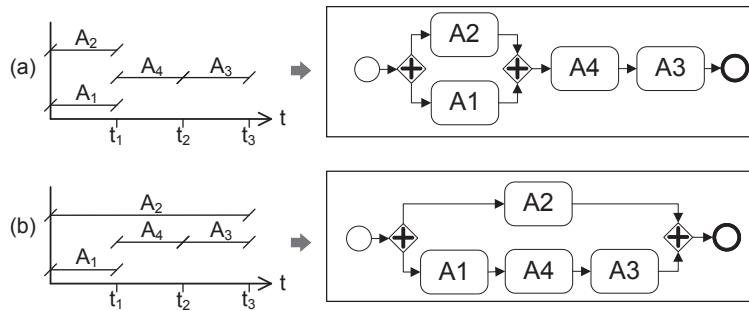


Figure 4.7 – Flexibility of the BP in terms of the execution time of the Activities

In this example, how the execution time of each activity can affect the process model flexibility is analysed. Figure 4.7 depicts an example formed of the activities A_1 , A_2 , A_3 and A_4 , where the inputs of A_4 depend on the output of A_1 , and the inputs of A_3 depend on the output of A_4 . Derived from the data dependencies, A_1 , A_4 and A_3 need to be executed sequentially, but the question remains of when A_2 can be executed. The execution options are: in parallel with each of the rest of the activities (for example in parallel with A_1 (as presented in Figure 4.7(a)) or in parallel with the sequence (as presented in Figure 4.7(b)). The model of Figure 4.7(b) is better than the model of Figure

4.7(a), since it allows activity A_2 to expend more time in the execution without increasing the total execution time of the process.

Therefore, in order to obtain the most parallelized process, all the possible values when an activity can finish should be known. To this end, we include in the COP the t_{ini} variables as a goal, and leaves the domain of t_{end} open during the search. This results in the t_{ini} variables being instantiated during the propagation phase, while all the possible values of the domains of t_{end} variables are returned when a value of t_{ini} is found. The greatest value of this domain is the most appropriate t_{end} for each activity.

Algorithm 1 describes the creation of the COP from a declarative specification. In particular, each activity and constraint of the declarative specification is transformed into a tuple $\langle t_{ini}, t_{end} \rangle$ (lines 3-6) and temporal constraint (lines 7-13) respectively. The t_{ini} of the activities are defined as goal variables to obtain a flexible and robust model (line 14). Finally, the minimization of the total time is defined as objective function in line 15.

Algorithm 1 Create a COP from Declarative Specification

```

1: Create variable:  $int TStart = 0$ 
2: Create variable:  $int TEnd = 0..number\ of\ Activities$ 
3: for each  $Activity(i)$  do
4:   Create variable:  $int t_{ini_i} = 0..number\ of\ Activities$ 
5:   Create variable:  $int t_{end_i} = 0..number\ of\ Activities$ 
6: end for
7: for each  $Activity(i)$  do
8:   Create constraint:  $t_{ini_i} \leq t_{end_i} + 1$ 
9:   Create constraint:  $TEnd \geq t_{end_i}$ 
10: end for
11: for each  $ADI_i = ADO_j$ , where an output of  $Activity\ j$  is related to the input of  $Activity\ i$  do
12:   Create constraint:  $t_{ini_i} = t_{end_j}$ 
13: end for
14: Define goal variables:  $\{ t_{ini_1} \cup t_{ini_n} \cup TEnd \}$ 
15: Define objective function:  $Minimize (TEnd - TStart)$ 

```

Using any one of the constraint programming solvers, a result with the instant of time in which each activity starts and ends can be obtained. Specifically, the solution of a COP is a list with the possible start time and end time of each activity. In the case of the t_{ini} , its value is returned directly from the COP, while the value of t_{end} is the greatest value of the domain returned from the algorithm.

4.3.2 Transformation of the COP results into a BP Imperative Model

The list of pairs for the t_{ini} and t_{end} for each activity, obtained from the COP, represent the most appropriate values for t_{ini} and t_{end} for each activity, but it has to be translated into a BP Model. Firstly, it is necessary to decide the standard modelling language in

which the BP is going to be represented. In our case, BPMN is applied since it is the most widely used standard language. As specified in (Weske, 2007) and (OMG, 2011b), BPMN is a graph-oriented language, therefore, we propose the creation of a directed graph from the obtained list that will represent a BPMN.

Our objective is to obtain a *BPMN – Graph* by analyzing the set of activities, and the times obtained from the COP. This *BPMN – Graph* represents a BPMN, obtained with Algorithm 2, that we explain in this section.

We define *ConfProblem* as the set of *activities* and a set of *relations*. The *relations* are established from the constraints specified in the declarative specification. Therefore, each *relation* is described by the tuple $\langle \textit{Condition}, \textit{Activity 1}, \textit{Activity 2} \rangle$, where *Condition* represents under which conditions *Activity 2* is executed after *Activity 1*. Therefore, the value *true* in the *Condition* implies that *Activity 2* is always executed after *Activity 1*, or, in the case when the *Condition* contains an expression, *Activity 2* is executed after *Activity 1*, if and only if the expression is met.

For the trip planner example, the input constraint (*IC6*)¹ could be represented as the *relation* $\langle \textit{DepartingFrom} \neq A_F.\textit{DepartingFrom}, A_F, A_{CR1} \rangle$, since the condition that establishes that *A_{CR1}* has to be executed after *A_F* is the value of *A_F.DepartingFrom*. With the activity time interval obtained from the COP, the information of the *ConfProblem* is completed by adding the $\langle t_{ini}, t_{end} \rangle$ information to each *activity*.

The proposed *BPMN – Graph* is a direct graph composed of: (i) *vertices*, which represent the activities and the gateways; and (ii) the *edges*, which represent the sequence-flow that join the various *vertices*. We also propose an algorithm to obtain an optimal *BPMN – Graph* in terms of the relationships of the activities.

Firstly, the idea of the algorithm is explained by means of a trace, after which the algorithm is detailed. The main idea of the algorithm is to apply a sequential and parallel treatments over the different parts of the *ConfProblem*. When these parts are transformed into *BPMN – Graphs*, they are combined into a larger whole. An example of the trace of the algorithm is shown in Figure 4.8. The *ConfProblem* is formed by a set of *activities* A_1, \dots, A_8 , where the $\langle t_{ini}, t_{end} \rangle$ of each *activity* (obtained from a COP resolution) and the *relations* are: (*R1*) $\langle A_1.\textit{output1} \leq 50, A_1, A_3 \rangle$; (*R2*) $\langle A_1.\textit{output1} < 100, A_1, A_4 \rangle$; (*R3*) $\langle \textit{true}, A_3, A_5 \rangle$; (*R4*) $\langle \textit{true}, A_4, A_5 \rangle$; (*R5*) $\langle \textit{true}, A_5, A_8 \rangle$; (*R6*) $\langle \textit{true}, A_2, A_8 \rangle$; and (*R7*) $\langle \textit{true}, A_6, A_7 \rangle$.

Firstly, the set of *activities* should be separated by the sequential points (arrow 1 in this example). A sequential point is an instant of time at which no activity is executed, and therefore the *ConfProblem* can be broken down into two smaller *ConfProblem*. In the example, the set of *activities* is separated into two subsets of *activities*, since there is only one sequential point in t_3 (there are no activities executing at this point). Each subset of *activities* is called a *ConfSubProblem*. Both *ConfSubProblems* are then analysed to establish the parallel subgroups of *activities* (arrows 2 and 3). The *secp1* is separated

¹As explained in subsection 3.2.1, the input constraint (*IC6*) was:

$$\begin{aligned} & \textit{departingFrom} \neq A_F.\textit{departingFrom} \rightarrow \{ A_{CR1}.\textit{departingFrom} = \\ & \textit{departingFrom} \wedge A_{CR1}.\textit{goingTo} = A_F.\textit{departingFrom} \wedge A_{CR1}.\textit{departDate} = \\ & A_F.\textit{departDate} \wedge A_{CR1}.\textit{returnDate} = A_F.\textit{returnArrivalDate} \} \end{aligned}$$

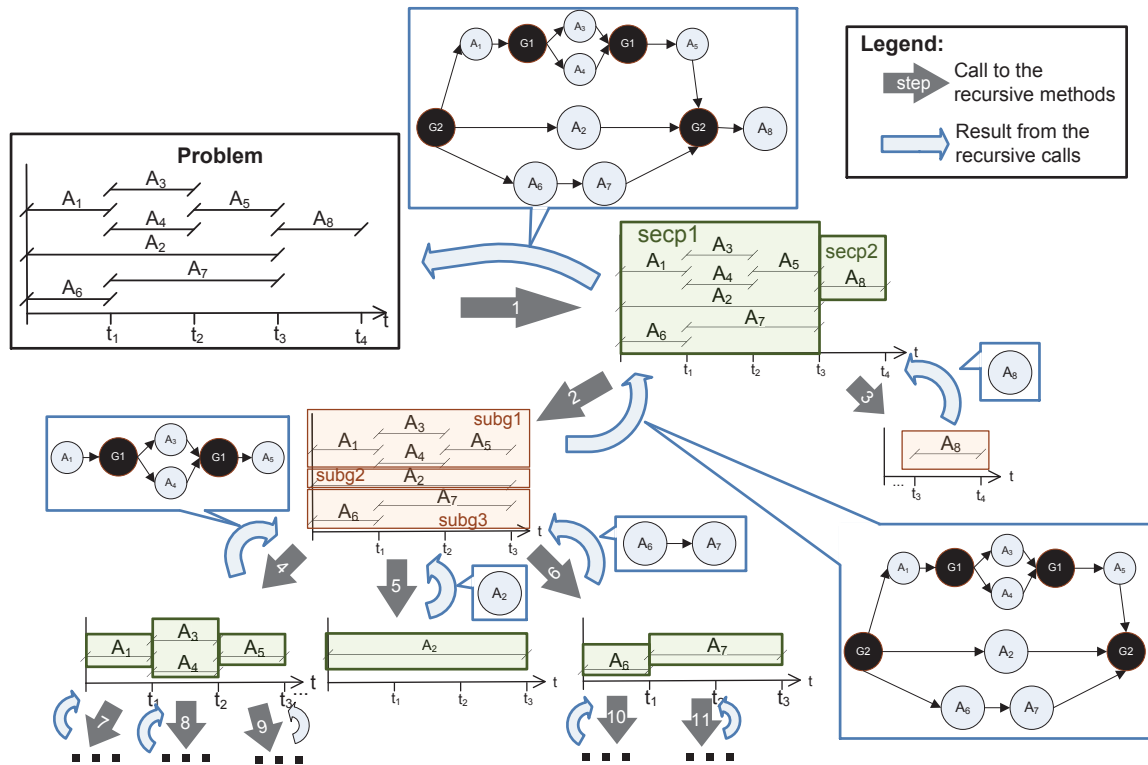


Figure 4.8 – Trace example of the Algorithm 2 to create a *BPMN – Graph* using the COP results

in three parallel subgroups: *subg1*, *subg2* and *subg3*, since, based on the *relations*, the *activities* with relationships are gathered. For example, due to the relations *R1*, *R2*, *R3* and *R4*, the activities *A1*, *A3*, *A4* and *A5* are gathered together with no other activities. The next step involves the sequential treatment of each subgroup (arrows 4, 5 and 6). The result of *secp1* (return of arrow 2) is a graph whose initial and final node is a gateway vertex, called *G2*, and the result of each *subgroup*: *subg1*, *subg2* and *subg3*, is linked to this vertex *G2*. On the other hand, the result of *secp2* is a graph with a vertex *A8* (return of arrow 3). Finally, the results of *secp1* and *secp2* must be joined together. To this end, the result (*BPMN – Graph*) of *secp1* and *secp2* are joined by an edge between the final vertex of *secp1* and the initial vertex of *secp2*, which are *G2* and *A8* respectively.

Algorithm 2 details the procedures for the sequential and parallel treatments explained above. Algorithm 2 takes a *ConfProblem* and transforms it into a graph. Initially, the *ConfProblem* is treated sequentially with the algorithm *sequentialTreatment* (lines 1-15), which involves breaking the problem down into sets of activities (*ConfSubProblem*) that are executed sequentially (line 8). These sets are differentiated for the existence of sequential points. If a sequential point differentiates two sets of activities (two *ConfSubProblem*), it means that there are no activities that can be executed in the intermediate point. Each of these *ConfSubProblem* has to be analysed to identify the

parallel sets of activities (lines 9-11) with the algorithm *parallelTreatment* (lines 16-30). The parallel treatment of a *ConfProblem* consists of gathering the activities that are related into groups and creating new problems from these groups (line 23). The relationships between the activities are given by the *relation* stored in the *ConfProblem*. Each *ConfSubProblem* is then treated sequentially as explained above (lines 24-26). Once all these *ConfSubProblem* are sequentially treated, as a result of the parallel treatment, a gateway vertex is inserted into the solution (graph), and all the solutions generated by the sequential treatment are linked to this gateway (line 28). Both treatments prevent activities of the *ConfProblem* from remaining divided into sequential *ConfSubProblem* or parallel subgroups. The algorithm draws to a halt when reaching a base case, or when the problem contains only one activity. In both treatments, the solution of a base case is a graph with a unique vertex: this activity (lines 5-6 and 20-21). On the other hand, two solutions are sequentially joined by an edge between the last node of the first solution and the first node of the second solution (line 13).

Once the graph is created, the correct gateways that diverge the sequence flows have to be selected. In the graph, the gateways are identified by means of studying the constraints, defined in the declarative specification, that may or may not be associated to the execution of these activities.

The conditions for each type of gateway are explained below:

- **Parallel:** two activities are executed in a parallel way, if there are no conditions that relate the t_{ini} of both activities (see Figure 4.9).

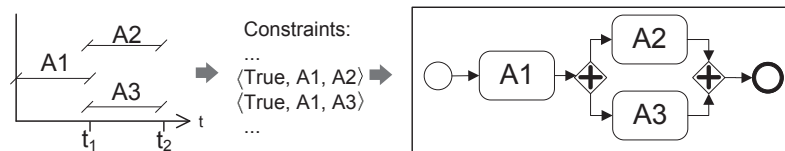


Figure 4.9 – Parallel Relationship

- **Exclusive:** two activities are executed in an exclusive way, if the constraints that relate the t_{ini} of the two activities and the domain of these constraints are complete and do not overlap. For example, as shown in Figure 4.10, if the execution of A_2 and A_3 depend on the value of the output data $A_1.oD1$, but there are no overlaps (A_2 is executed when $A_1.oD1$ is less than 50 and A_3 is executed when $A_1.oD1$ is greater or equal to 50), then there is an exclusive relationship between them.

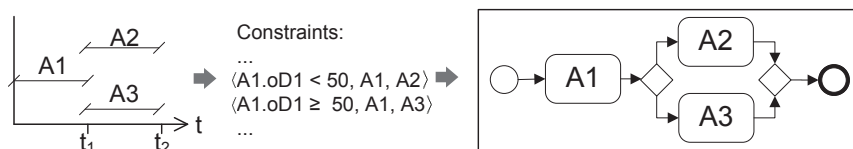


Figure 4.10 – Exclusive Relationship

Algorithm 2 Create a BP model from COP result

```

1: procedure SEQUENTIALTREATMENT(ConfProblem p): BPMN-Graph
2:   sol: BPMN-Graph to return
3:   lprob: list of ConfProblem
4:   lsol: list of BPMN-Graphs
5:   if (p.listActivities.size() == 1) then
6:     sol ← p
7:   else
8:     lprob ← separate p by sequential points
9:     for each p ∈ lprob do
10:      lsol.add(parallelTreatment(p))
11:    end for
12:  end if
13:  sol ← link sequentially lsol
14:  return sol
15: end procedure
16: procedure PARALLELTREATMENT(ConfProblem p): BPMN-Graph
17:   sol: BPMN-Graph to return
18:   lprob: list of ConfProblem
19:   lsol: list of BPMN-Graphs
20:   if (p.activitiesList.size() == 1) then
21:     sol ← p
22:   else
23:     lprob ← separate p in parallel groups
24:     for each p ∈ lprob do
25:      lsol.add(sequentialTreatment(p))
26:    end for
27:  end if
28:  sol ← link lsol by gateways
29:  return sol
30: end procedure

```

- Inclusive:** two activities are executed in an inclusive way, if there are conditions that relate the t_{ini} of both activities, and the domain of these conditions are complete and overlapping. For example, as shown in Figure 4.11, if the execution of A_2 and A_3 depends on the value of the output data $A_1.oD1$, but there are overlaps between the domains that satisfy the constraints (A_2 is executed when $A_1.oD1$ is less than 75 and A_3 is executed when $A_1.oD1$ is greater or equal to 25, and hence both coincide when $A_1.oD1$ is greater than 25 and less than 75), then there is an inclusive relationship.



Figure 4.11 – Inclusive Relationship

Table 4.2 gives a summary of the resulting gateways depending on the conditions established by the constraints that relate the data of the activities.

Gateway	Constraints	Conditions
Parallel	No dependencies in domains	
Exclusive	Domains without overlaps	
Inclusive	Domains with overlaps	

Table 4.2 – Type of gateway decision

For the example explained in Figure 4.8, it is necessary to analyse $G2$ and $G1$ (see Figure 4.12). Since there are no relations between activities A_1 , A_2 and A_6 , then $G2$ is a parallel gateway. On the other hand, the execution or not of activities A_3 and A_4 depends on the outputs of A_1 . The conditions specified in relations $R1$ and $R2$ indicate that the domain of the constraints is complete and there are no overlaps, therefore, $G1$ is an exclusive gateway.

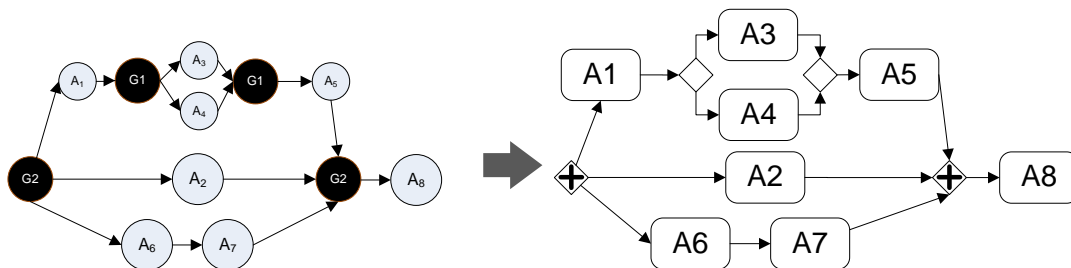


Figure 4.12 – From graph to BPMN Model Example

4.4 Configuration applied to the Trip Planner

In order to illustrate the use of the algorithms, the application of the algorithms to the Trip Planner case study is shown in this section. The input for the configuration problem is the declarative specification given in Chapter 3. The Trip Planner is formed by a set of data with the customer requirements, three activities (Flight, Hotel, and Car Rental Search Activities), a set of constraints that relate the data, and an objective function to be optimized: the minimization of the total price of the trip. The problem was also represented using DOODLE, and included in a BPMN model as a CombA Sub-Process.

Given this declarative description, the first step is the Trip Planner transformation to a COP definition by applying Algorithm 1. Each activity has an associated tuple $\langle t_{ini}, t_{end} \rangle$ as shown below:

- Flight Search Activity: $\langle TFlight_{ini}, TFlight_{end} \rangle$
- Hotel Search Activity: $\langle THotel_{ini}, THotel_{end} \rangle$
- Car Rental Search Activity: is divided into two different activities since there are two types of rental cars. Both rental cars are defined as $\langle TCarRental1_{ini}, TCarRental1_{end} \rangle$ and $\langle TCarRental2_{ini}, TCarRental2_{end} \rangle$.

As the execution time for each activity remains unknown, we consider that every activity lasts one unit of time. Therefore, for each activity, t_{ini} takes a domain with values between 0 and 3 ($numberOfActivities - 1$) and t_{end} takes a domain with values between 1 and 4 ($numberOfActivities$). In the worst case scenario, all the activities are executed sequentially, and the total time is equal to the number of activities: 4. In the best case, every activity is executed in parallel and the total time is 1.

The data dependencies are then transformed into temporal constraints. For example, the constraint which establishes that if the flight does not serve the destination city, then the rental of a car is necessary. This data dependency generates the constraint $\{TCarRental2_{ini} \geq TFlight_{end}\}$ since it is necessary to know the output data of Flight Search Activity in order to determine the input data of Car Rental 1 Search Activity.

The result obtained indicates a value or values assigned to each variable. The variables that can have various values are the end times of the activities (t_{end}), this means that if a variable has a domain (various values), it is due to the flexibility of this activity in that it can finish at different moments. The result for the trip planner example is:

- Goal Variables:
 - $TFlight_{ini}$ value equal to 0
 - $THotel_{ini}$ value equal to 1
 - $TCarRental1_{ini}$ value equal to 0
 - $TCarRental2_{ini}$ value equal to 1

- Objective Function:
 - minimize (T_{End}), value equal to 2
- Rest of Variables:
 - $T_{Flight_{end}}$ value equal to 1
 - $T_{Hotel_{end}}$ value equal to 2
 - $T_{CarRental1_{end}}$ values between 1 and 2
 - $T_{CarRental2_{end}}$ value equal to 2

According to Algorithm 1, from the remaining variables, we take the highest value of the set of possible values. In the example, the variable with a set of possible values is only $T_{CarRental1_{end}}$, which finally takes the value 2.

The resulting BP Model, obtained from applying Algorithm 2, is shown in Figure 4.13. According to the results, “Flight Search” and “Car Rental 1 Search” activities start at instant 0. However, the “Car Rental 1 Search” activity depends on the value of the input data of the “Flight Search” activity, so there is an exclusive gateway before this activity. Since it is not the complete domain with the constraint that determines the execution of the “Car Rental Search 1” activity, then there is a branch by default which indicates that nothing happens when the condition is not met. The same situation occurs with “Hotel Search” and “Car Rental 2 Search” activities.

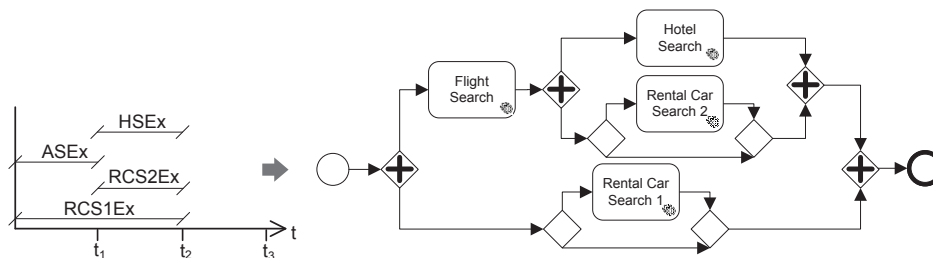


Figure 4.13 – Trip Planner Model Result

4.5 Related work

The use of Constraint Programming in configuration problem is very frequent, since the configuration involves selecting and arranging parts to achieve particular goals with respecting particular constraints. The use of CSP to solve configuration problems has been using in several works, from the management of configurations in self-adaptive systems (Sawyer et al., 2012) until service composition (Thiagarajan and Stumptner, 2007).

Configuration problems have also been used in BPs, but they are applied to obtain BP model families, by means of capturing all the possible BP that can be defined for a target domain of interests (Gröner et al., 2013). However, this family of BP comprises a variability modelling perspective or model template which does not fit with the objective of data-oriented optimization problems. Furthermore, these approaches represent this variability, by means of feature model analysis, or the application of software product lines techniques (Dhring et al., 2014; Ognjanovic et al., 2012; Rosa et al., 2011).

In (Runte, 2009), the author exposes the ideas of applying Constraint Programming to ensure the consistency in sequential and hierarchical relations of BP, which is then developed in (Runte and Kharbili, 2009). Although the authors expose the importance of the data-dependencies between various BP, a BP is not created at design time based on these BP requirements. They found out the inconsistencies at runtime and then, proposed configurations of the BP where these inconsistencies do not appear. A similar approach is proposed in (Borrego, 2012), where Constraint Programming is applied to verify the correctness of a BP according to a model represented by means of constraints, but in that work how to modify the model to satisfy the constraints is not proposed.

To the best of our knowledge, none of the existing proposals present a configuration solution to build an imperative BP model, based on data-structure dependencies represented declaratively.

4.6 Summary and Discussion

In this chapter, a configuration system to establish an optimal imperative BP model according to data-structure dependencies has been proposed. The main limitation for the business experts is the explicit representation of a process when they only know the BP requirements (activities, data, constraints, and objective function), but not how to represent these requirements in an imperative model. The configuration of an imperative model from the declarative description is a difficult and hard task, since the flexibility and adaptability that only a declarative description can offer must be maintained. Moreover, when the flexibility depends on the dependencies of activities through their input and output data relationship, the analyses of every possible configuration is even greater.

The Configuration System presented in this chapter establishes a methodology and a set of algorithms to configure an imperative representation from a declarative model. Business experts only have to specify the BP requirements, leaving the configuration of the activities in the imperative model to the system, which uses the Constraint Programming paradigm and a set of algorithms. This automatic method obtains an imperative model described by BPMN, where the optimal model related to minimize the possible execution time of the instances is obtained. The order between the activities is obtained by analysing the data-structure dependencies, by means of studying the constraints defined in the declarative specification. These constraints could establish the sequential, parallel, inclusion and/or exclusion execution structure between the activities. Finally, the configuration is applied to the trip planner case study introduced in Section 2.6 .

Chapter 5

Creating an Imperative Model to Optimize the Business Process Outcome

Good programmers use their brains, but some good guidelines save us having to do it in each case.

Francis Glassborow

5.1 Context and Motivation

One of the main challenges of a data-oriented optimization BP is the necessity to check every possible combination of data with the aim of obtaining the optimal outcome of the BP, e.g. tried every combination of dates and cities to obtain the cheapest trip. Thanks to the declarative specification proposed in Chapter 3, the BP requirements focused on data can be described and represented shielding the business expert to know how is the imperative model or the specific values for the variables that optimize its requirements. These requirements establish the set of data, activities, and constraints that related the activities through the data, and the objective function to be optimized. However, this declarative specification is not enough to be executed neither deployed in a commercial BPMS. In previous chapter, how to transform the declarative model with data-structure dependencies to an imperative model was necessary. But how the data values can affect the optimization of the outcome needs to be studied as well. In this chapter, how the imperative model obtained in the previous chapter must be modified to support the data optimization is analysed. The goal of the contribution presented is to maintain the flexibility of the declarative specification by delegating the search of the optimal data combination to a component included in the imperative model. For example, in order to know the flight, hotel, and car rental prices (output of the corresponding search activity), the specific values for the input data must be known to execute each activity.

This chapter is focused on the design of the functionality in charge of supplying the

search of all possible combination of input data in order to execute the activities. This functionality depends on the knowledge of the relationship between input and output data of the activities, specifically, if this relationship can be known with or without executing the activities. For the Travel Search sub-process, the objective function is to minimize the *totalPrice*, which can only be known when the activities are executed in runtime. However, if the pre- and post-conditions of the activities could relate input and output data, then the output values of the activities, for given input values, can be obtained without the execution of the activities. This implies the possibility of solving the optimization problem in a local manner, thereby reducing the complexity and the way in which the problem is solved in an imperative model. Therefore, regarding the knowledge about the behaviour of the activities described by the pre- and post-conditions, two different imperative models could be created:

- **White-Box Model:** This is used when the problem is formed by *white-box* activities, that means that the output values of the output variables can be obtained in terms of the input values, the pre-, and the post-conditions, and it is unnecessary to execute the activity to ascertain the output of the activity. In this case, the optimization problem can be modelled and solved in a local manner, since the execution of the activities is unnecessary to ascertain the values of the output data for each activity which optimize the objective function.
- **Black-Box Model:** This is used when the problem is formed of *black-box activities*, which implies that the pre- and post-conditions of the activities are insufficient to ascertain the output values, given a set of input values. In this case, it is necessary to execute the activity to obtain the specific values for the data output, since the model is unknown in a local manner. However, the creation of all the possible combinations of values for the data input can be modelled and solved locally. These combinations of input data values generated are used in the execution of the activities (cf. Chapter 4) to ascertain the specific values for the objective function to optimize.

Using the imperative model obtained in the previous chapter, we propose to use constraint programming and a set of algorithms to manage the possible data values that optimize the outcome of the business process. Both white-box and black-box specification can use this paradigm to find the best input data combination, where the main difference is how the objective function is managed. The objective function can be included directly in the COP, and solved it locally (white-box); or managed depending on the output values of the activities once they are executed given a combination of input data obtained from a CSP (black-box). Although both solutions give reasons for the necessity to construct a COP and a CSP, the following subsections focus on how white-box and black-box models develop the necessary functionality to support the data-search in an imperative business process.

The remainder of the paper is organized as follows: Section 5.2 and Section 5.3 explain how white-box and black-box models can modify the imperative model to support the declarative model according to data-value dependencies. Finally, conclusions are drawn and future work is proposed in Section 5.5.

5.2 The White-Box Approach

The white-box models are used when the activities that form the data-oriented optimization process are white-box activities. A white-box activity is an activity that giving a set of values of the data input that satisfy its pre-condition, the values of the data output can be known without the execution of the activity, only by analysing the post-condition of the activity.

Execution of the activities to optimize the objective function is avoided by transforming the declarative model into a generic COP capable of support any instance. We propose that this COP can be set-up as a script activity within the imperative model, created in previous chapter, to manage the possible values of the input data variables of the activities. Once an instance arrives to this script activity, the variables of the COP are instantiated with the concrete values, and with the use of a solver integrated in the BPMS, this COP can be solved, given as a result the optimal outcome.

Therefore, determining and inferring an executable activity by means of automated transformation from a data-oriented optimization declarative specification is proposed, as depicted in Figure 5.1. To this end, it is necessary: (1) to create automatically a COP according to the data-oriented optimization declarative model; and (2) to create a script activity with the resulting COP and incorporate it into the imperative model. The COP is solved at instantiation time in terms of the customer requirements for each instance.

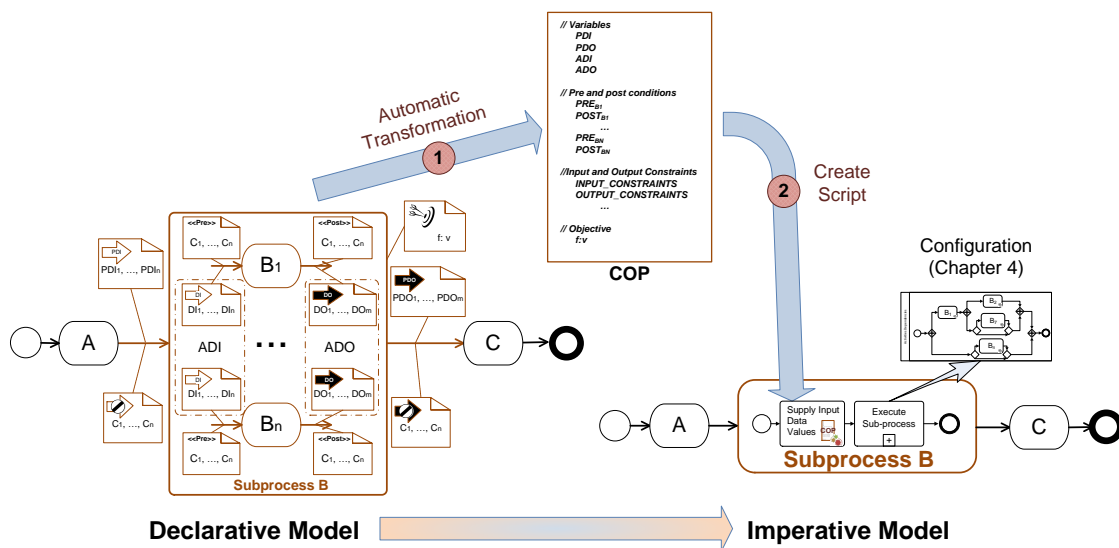


Figure 5.1 – White box transformation.

5.2.1 How to transform a White-Box Specification into an Executable Business Process

In order to design a model that involves all the variables and constraints that describe the data-oriented optimization process, the following COP structure is proposed:

- X : DF, which implies the PDI, PDO, `DATA_INPUT(A1)`, ..., `DATA_INPUT(An)`, `DATA_OUTPUT(A1)`, ..., `DATA_OUTPUT(An)`.
- D : Domain depending on the DF variables. The values of the domain depend on each instance of the BP.
- C : `PRE(A1)` \wedge `POST(A1)` \wedge ... \wedge `PRE(An)` \wedge `POST(An)` \wedge `INPUT_CONSTRAINTS` \wedge `OUTPUT_CONSTRAINTS` \wedge {Instantiation of the PDI with the values of the specific instance}
- O : `OBJECTIVE_FUNCTION`

The incorporation of the COP in the imperative model allow to obtain at runtime, the most appropriate input data for the activities to optimize the objective function. The creation of the COP is carried out using a model-to-text transformation. In our case, Epsilon (Foundations, 2014) has been used. Among other things, Epsilon provides a family of languages and tools for code generation and model-to-model transformation.

Once the COP is built, it must be set-up as a *script activity* (OMG, 2011b) within the imperative model obtained after the configuration process. This imperative model can then be executed in any of the existing BPMSs. Therefore, the creation of this script activity depends on the mechanisms provided by the modelling tool of the BPMS used, and also on the BPMS engine capabilities.

Regardless of the BPMS used, the COP can be solved using any of the existing COP solvers, Choco Solver (Choco Solver, 2014) in our case, which connects the script activity with an external constraint programming solvers.

In our case, the BPMS employed to design and execute the BPs is Bonita Open SolutionTM since it is an open-code application with a free distribution, and is commonly used in the private market. We have implemented a connector called *COP Script* in order to solve the COP inside the BP. A *connector* is the way in which Bonita Open SolutionTM (Community, 2012a) links an activity with a service or application that executes a functionality. The connector has been implemented with the libraries provided by Choco (Choco Solver, 2014) solver, and the plug-in of Bonita Open SolutionTM to extend its functionality. The business modeller can therefore use this connector to link the COP solver with the script activity. Figure 5.2 shows an example of how a business modeller can include the generated COP into any script activity using the developed *COP Script connector* in Bonita Open SolutionTM (Community, 2012a).

Finally, at runtime, when an instance reaches the script activity with the specific data values, the COP is solved and the result is obtained.

5.2.2 White-Box Model Transformation applied to the Trip Planner

A clear example of white-box transformation in the trip planner example occurs when the cost of each part of the travel is included in different travel brochures, such as the offers

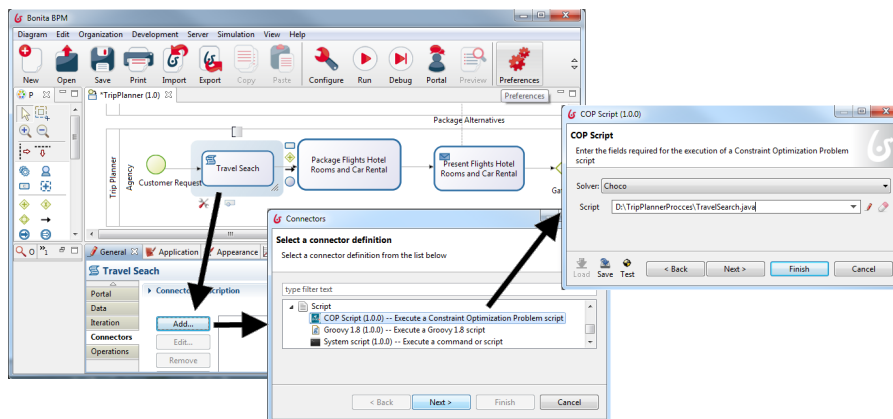


Figure 5.2 – Choco Script Task Configuration in Bonita Open SolutionTM.

available at agencies. In this case, since the prices are pre-established for specific date and conditions, the sub-process knows, without executing the activities, the data input and output relation for each part of the trip.

Therefore, the COP generated from the model specified in Chapter 3 is shown in Table 5.1. Since we are analysing white-box models, the relation between input and output are known. For example, all flights from Seville to London, regardless of the date, costs 100 euros in November ($A_F.departureDate \geq \text{“2014-11-01”} \wedge A_F.departureDate \leq \text{“2014-11-31”} \rightarrow priceFlight = 100$). Since the travel brochures are accessible and public, and cannot change for the dates published, then this information can be included in the COP.

```
//Variables:
  earlyDepartureDate: Date
  lastDepartureDate: Date
  priceFlight: Integer
  ...
//Pre and post-conditions
  AF: AF.departureDate > SystemDate
  AF.departureDate ≥ “2014-11-01” ∧
  AF.returnDate ≤ “2014-11-31” → priceFlight = 100
  ...
  AH: AH.checkOutDate > AH.checkInDate
  AH.checkOutDate > AH.checkInDate
  ...
// Input and Output Constraints
  AF.returnDate = AH.checkOutDate
  earlyDepartureDate ≤ AF.departDate
  AF.departDate ≤ lastDepartureDate
  ...
//Optimization function:
  TotalPrice = priceFlight + priceHotel + priceCarR1 + priceCarR2
//Objective
  minimize(TotalPrice)
```

Table 5.1 – Constraint Optimization Problem for outcome data optimization

5.2.3 Empirical Evaluation

In this section, the experimental results corresponding to the evaluation performed are shown, the obtained results are discussed, and the limitations of our proposal are laid out.

5.2.3.1 Experimental Design

Since the generated COP only depends on the declarative model, it is created only once in design time. Therefore, this COP is used for every instance at runtime, and finds the best option for any customer requirements and maintains the flexibility. The COP provides the necessary mechanisms to obtain the optimal value of the process by checking the various existing catalogues. Therefore, the critical phase is on the resolution of the COP at runtime. A faster resolution of the COP implies that the desired result is obtained earlier.

In general, the time to find the solution of a COP depends on: (1) the type of constraints (lineal, polynomial); (2) the number of constraints; and (3) the number and type of the variables. It is possible to find an analysis in (Cheeseman et al., 1991) on the complexity of the NP-complete constraint problem resolution according to these characteristics. The complexity of resolution of CSPs depends on the number of possible solutions of the problem, and on whether it is neither under constraint nor over-constrained. The most complex of these problems are those that are neither under constraint nor over-constrained. For these reasons, no affirmation can be given concerning the efficiency in a generic way of our proposal, since our declarative specification permits any type and any number of constraints; therefore the evaluation time depends on the specific problem. Depending on the number of constraints associated to a COP, and the number of variables, the COP evaluation time remains variable.

For this reason, and with the aim of performing the evaluation of our proposal, the resulting COP of the case study is executed over a set of 150 generated test cases. Each test case establishes a different number of values for the PDI. For their part, the different values for each PDI generate, in turn, a number of combinations for PDI variables, in such a way that each test case increases the number of combinations as shown in Figure 5.3. Since not all the combinations are possible solutions of the problem, Figure 5.3 also indicates the number of possible solutions for each test.

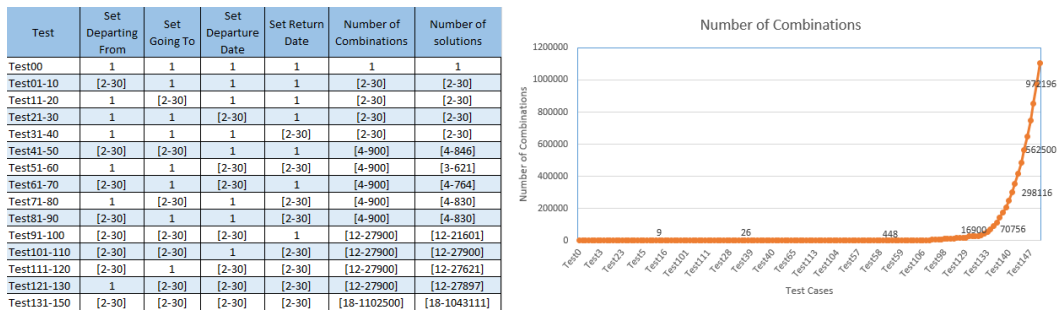


Figure 5.3 – Number of possible combinations for each test case.

5.2.3.2 Experimental Result

Therefore, the main purpose of the experimental evaluation is the determination of the computing time(s), the memory used (MB) and the quotients between the time and used memory needed to solve the COP of the example. In order to realise the importance of the optimization, the measurements have been carried out over the COP with and without the objective function, and hence, all the solutions are found.

The test cases were measured using a PC with an Intel Core i7-2675QM CPU with a 2.2 GHz processor and 8.00 GB of RAM.

Figure 5.4 shows the memory used to solve the COP. It is possible to notice that, for the *optimized* option, the memory used is almost constant regardless of the number of permutations. On the one hand, the used memory of *all solution* option is exponential.

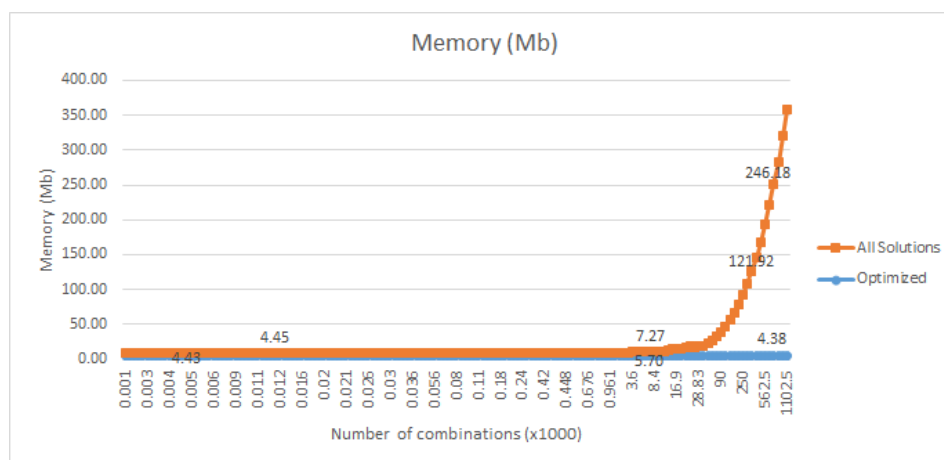


Figure 5.4 – Memory used for the tests of the example.

On the other hand, the behaviour of the results of time fits an exponential model for *all solution* option. However, it is necessary to highlight the increase in time for the *optimized* option. However, it is necessary to highlight the increase in time for the optimized option, which remains almost constant, in comparison with the great increase with respect to the number of combinations (see Figure 5.5).

5.3 The Black-Box Approach

The black-box models are used when the activities that form the data-oriented optimization process are black-box activities. A black-box activity is an activity that giving a set of values of the data input that satisfy its pre-condition, the values of the data output are unknown until the activity are executed, which means that the pre- and post-conditions of the activities are insufficient to ascertain the output values, given a set of input values.

The main problem of black-box activities is that until the activities are not executed, their output are unknown. This implies that the activity in charge of supplying the input data to the imperative model, obtained in Chapter 4, must ensure that the activities not only carry out an instantiation of the variables that satisfies all constraints, but also

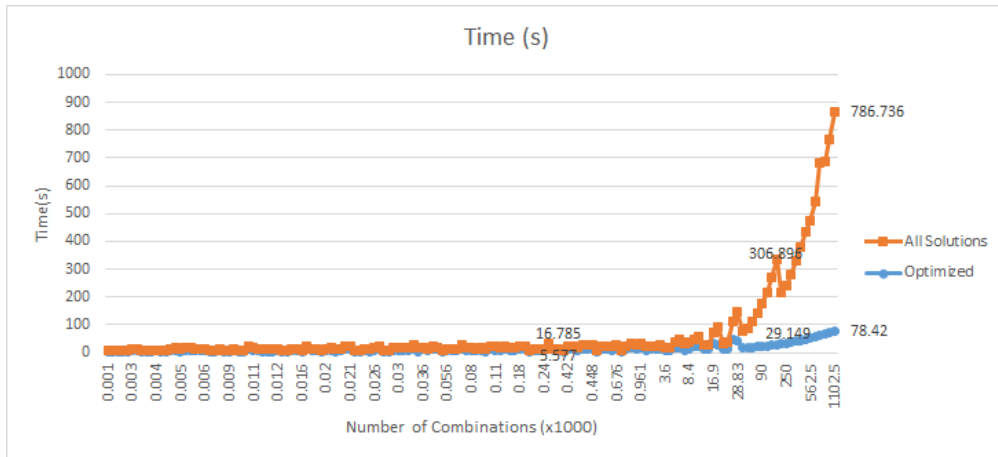


Figure 5.5 – Time used for the tests of the example.

searches for the optimal value of the objective function. In other words, now the functionality of this activity must be the coordination of the search. Therefore, a coordinator activity, called *Coordinator* is necessary (see Figure 5.6).

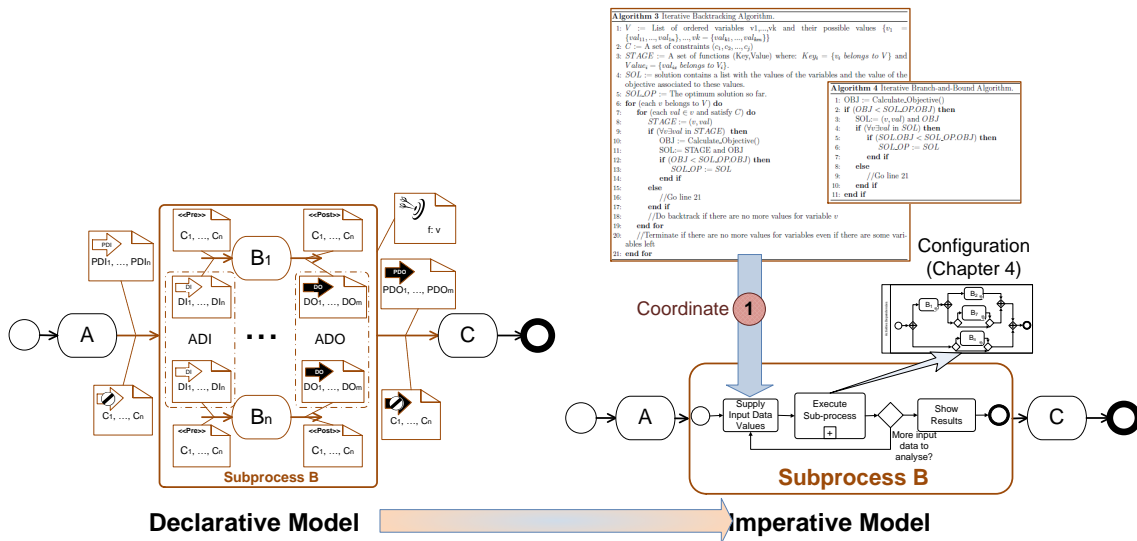


Figure 5.6 – Black box approach schema.

This *Coordinator* is an script activity which has the knowledge of the whole problem: from the objective function to the accesses to the activities needed. In addition, the *Coordinator* is also responsible for executing the algorithm necessary for the assignment of values to variables. Each activity has certain ADI; therefore the coordinator must provide such data to each activity. Therefore, the *Coordinator* is composed of an algorithm that instantiates the variables and execute the activities to obtain the values necessary for the calculation of the objective function. Furthermore, it must prevent repetition of the searches since it could carry out endless loops. Following subsections detail the algorithms developed for this coordinator and the experimental results done.

5.3.1 Coordinator Algorithm

The algorithm used by the coordinator has to consider all possibilities and retains only the best result. Algorithm 3 is based on DisCSP algorithms, which can be classified as a combination of Centralized (Yokoo and Hirayama, 2000) and Synchronous Backtracking (Yokoo et al., 1998). Centralized, since there is an *Coordinator* which has overall knowledge of the problem: organizing the different activities, taking control of the instantiation of variables, and ensuring that the activities achieve the objective function. It is also synchronous since all activities remain in communication only when the *Coordinator* needs to combine their returned values.

In the same way as for the search space in backtracking algorithms, the search space in Algorithm 3 also has a tree structure. The *Coordinator* chooses the variable that is instantiated at each level of the tree, by composing the new partial candidate. The various activities return the best solution according to the values of the instantiated variables and the ranges of the non-instantiated variables.

Algorithm 3 Iterative Backtracking Algorithm.

```

1:  $V :=$  List of ordered variables  $v_1, \dots, v_k$  and their possible values  $\{v_1 = \{val_{11}, \dots, val_{1n}\}, \dots, v_k = \{val_{k1}, \dots, val_{km}\}\}$ 
2:  $C :=$  A set of constraints  $(c_1, c_2, \dots, c_j)$ 
3:  $STAGE :=$  A set of functions (Key, Value) where:  $Key_i = \{v_i \text{ belongs to } V\}$  and  $Value_i = \{val_{ix} \text{ belongs to } V_i\}$ .
4:  $SOL :=$  solution contains a list with the values of the variables and the value of the objective associated to these values.
5:  $SOL\_OP :=$  The optimum solution so far.
6: for (each  $v$  belongs to  $V$ ) do
7:   for (each  $val \in v$  and satisfy  $C$ ) do
8:      $STAGE := (v, val)$ 
9:     if  $(\forall v \exists val \text{ in } STAGE)$  then
10:       $OBJ :=$  Calculate_Objective()
11:       $SOL := STAGE$  and  $OBJ$ 
12:      if  $(OBJ < SOL\_OP.OBJ)$  then
13:         $SOL\_OP := SOL$ 
14:      end if
15:    else
16:      //Go line 21
17:    end if
18:    //Do backtrack if there are no more values for variable  $v$ 
19:  end for
20:  //Terminate if there are no more values for variables even if there are some variables left
21: end for

```

The key of the algorithm is in line 10, where the function that calculates the objective of the problem is invoked (*Calculate_Objective()*). This function is the execution of the

BP which contains the imperative model configured in Chapter 4 in order to obtain the output values.

In general, the objective of a backtracking algorithm is to find the best value according to a certain criteria over the set of solutions. In order to reduce the search space, a branch of the search tree is discarded (pruned) if a node is reached (partial candidate), and if it is known that this branch of the tree will not improve the best solution for $f(x)$ found so far. Thus, once the algorithm obtains a first solution, if the function *Calculate_Objective()* fails to return a value that can improve this solution, then this branch of the search is pruned. Following on with the Trip Planner example, the customer wants to buy an airline ticket to travel from Seville to London and to book a hotel in London. Moreover, to obtain the cheapest trip, the customer is flexible with the departure date: 2014-07-01 or 2014-07-02 and return date: 2014-07-05 or 2014-07-06. The Coordinator executes Algorithm 3 (Backtracking Iterative Algorithm) and obtains the best solution. The resulting tree structure is shown in Figure 5.7.

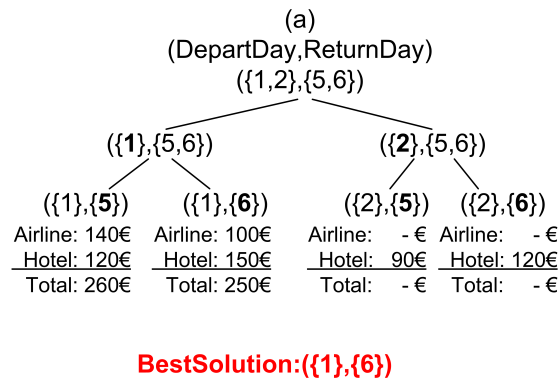


Figure 5.7 – Backtracking Tree Structure.

The function *Calculate_Objective()* is called once there is a complete assignment. However, the function *Calculate_Objective()* replies that there are no flights departing on 2014-07-02 and the return date is either 2014-07-05 or 2014-07-06, therefore, why call *Calculate_Objective()* giving all the possible combinations with dates, if it is impossible to obtain a better solution since there are no flights? (see Figure 5.8).

The new algorithm modifies Algorithm 3 from line 9 to 17, as shown in Algorithm 4. The bound is applied before setting the value of all variables to further reduce the search space.

5.3.2 Experimental Results

The empirical evaluation of the techniques is focused on performance measures of the algorithms presented. These algorithms can be applied to any BP that has to achieve the optimization. The main purpose of the experimental evaluation is the determination of the time needed to find the optimal: the function *Calculate_Objective()* is responsible for this objective since it invokes the BP that calls the activities to combine. Therefore, estimating

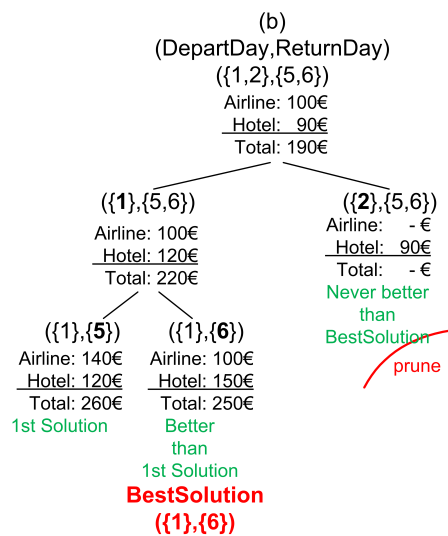


Figure 5.8 – Branch and Bound Tree Structure.

Algorithm 4 Iterative Branch-and-Bound Algorithm.

```

1: OBJ := Calculate_Objective()
2: if (OBJ < SOL_OP.OBJ) then
3:   SOL := (v, val) and OBJ
4:   if (∀v∃val in SOL) then
5:     if (SOL.OBJ < SOL_OP.OBJ) then
6:       SOL_OP := SOL
7:     end if
8:   else
9:     //Go line 21
10:  end if
11: end if

```

the number of calls to *Calculate_Objective()* is the same as calculating the time to obtain the optimal outcome. Thus, the number of calls to the function *Calculate_Objective()* are calculated in both algorithms for comparison purpose. On the other hand,

The hardware used in the execution test is a server Intel Xeon 2.40GHz - 8GB RAM where the Web Services are located, and an Intel Core 2 Duo 3.00GHz - 3.49GB RAM, where test cases are measured.

For the evaluation, the algorithms are applied to the problem based on the illustrative example, the Trip Planner. The Flight Search Activity and the Hotel Search Activity internally use a database (DB) to establish the price and the Rental Car Search Activity uses input values and a set of constraints to establish the price, and hence can be described as a simple CSP.

A comparison of the execution of these two algorithms is carried out over a set of test cases. In order to create an effective and efficient comparison, each test case is composed of different values and combinations of input parameters. These values and combinations

are sufficiently representative to perform a good comparison between the two algorithms.

Both algorithms obtain the same price in each test case of the airline ticket, the hotel and the rental car for the same dates. However, the most interesting parameter is the number of calls to the function *Calculate_Objective()* in both algorithms (Figure 5.9).

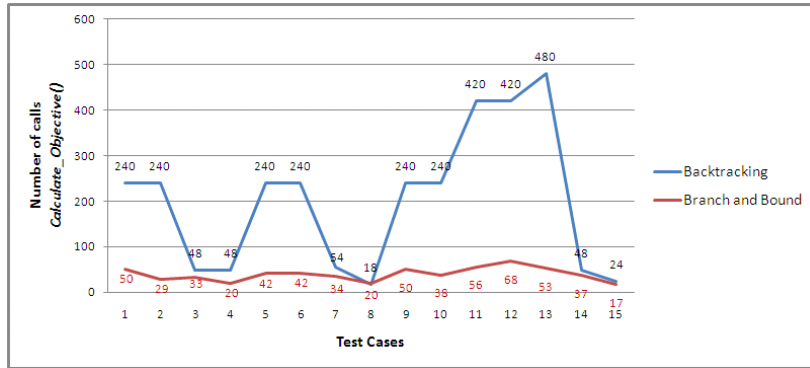


Figure 5.9 – Number of calls to *Calculate_Objective()* by Algorithm 1(*Backtracking*) and by Algorithm 2 (*Branch-and-Bound*).

The difference between the number of calls by the Backtracking algorithm and by the Branch-and-Bound algorithm is outstanding. In most cases, the number of calls in the Branch-and-Bound solution is lower than that of the Backtracking algorithm, except for Case 8, where the Backtracking calls 18 times and Branch-and-Bound 20 times. If there is a good bound and the best price is at the beginning of the search space, it will not always be necessary to call *Calculate_Objective()* in every case. However, this does not occur in Case 8. This case highlight the importance of good bounds, since it can reduce considerably the search space. Therefore, the number of calls to the *Calculate_Objective()* is not only important, but also the extra processing that has the branch-and-bound to not make so many calls, which means a tradeoff between both factors.

5.4 Related work

In general, papers found in the literature related to the transformation from declarative to imperative descriptions are not focused on the assistance of the customer for the input data. Only in (Gómez-López et al., 2011) is the decision-making support oriented towards data input, but it omits optimization of an objective function, and only makes the BP instances satisfiable for imperative models. In other work related to how to model the processes, such as (Barba et al., 2013), Barba et al. propose a framework of assistance to create models which take the necessary resources involved in the process into account. In that paper, the data that describe the resources of the execution of the process are used, but not the data that flow at runtime, nor does it consider the input data for the activities that constitute the process. On the other hand, there are various studies that apply transformations to BP modelling. Victoria Torres et al. in (Torres and Pelechano, 2006) apply transformations to generate the navigation between web pages from a BP

definition, specifically the BPEL executable description that implements the entire process. In (Fabra et al., 2012), Fabra et al. integrate a service-oriented development method (SOD-M) and a platform for the development and execution of interoperable dynamic web processes (DENEb). They present a model-driven framework for the analysis, design, development and execution of business processes which covers BPM solutions from the very early stages of their development to their deployment and execution. Furthermore, an existing gap is highlighted between BPM and Service-Oriented Architecture, and several solutions that use Model-Driven Architecture to cover this gap are also pointed out. Although a certain number of these papers are focused on defining the interaction between various participants in order to achieve business goals, none of them deals with the type of problems presented in this work: the creation of an imperative model to minimize the BP execution time focused on data dependencies.

5.5 Summary and Discussion

In this chapter, we present two approaches focused on designing the necessary mechanisms to check every possible combination of data of a data-oriented optimization BP. The proposals show how the declarative language is automatically transformed into an executable process. This executable activity is in charge of obtaining the outcome data optimization of each instance of the process for customer requirements. Firstly, our focus is on problems formed by *white-box* activities, which means that the output values of the output variables can be obtained in terms of the input values by means of the pre and the post-conditions, thereby rendering the execution of the activity unnecessary in the search for knowledge of the output of the activity. Thus, the execution of the activities for the optimization of the objective function is avoided by incorporating the imperative model that support the data-structure dependency of the activities into a script activity with a COP. The use of the constraint programming paradigm enables the necessary input data values that optimize this outcome data to be ascertained. The approach presented is detailed and tested with a case study implemented in a commercial BPMS.

Secondly, our focus is on problems formed by *black-box* activities, which means that the activities have to be executed in order to know the output values of the output variables given a set of input values. In order to solve the aforementioned problem, an adaptation of a Backtracking algorithm is developed together with an improvement using Branch-and-Bound algorithm. This adaptation permits the activities to obtain the optimal outcome, which, in the case of the trip planner example, is the determination of the cheapest trip. One of the main problems in the search for solutions is the size of the search space. A bound is established to prevent unnecessary executions of activities when it is known that no better result could be obtained. Results show that a well-designed bound can significantly reduce the performance time of the algorithm.

Part V

Contributions III: PAIS-DQ

Chapter 6

PAIS-DQ

It is your design, it is your decision.
Anonymous.

6.1 Context and Motivation

The BPs under the scope of our research are those which are centred on developing sound data in business processes, analysing how data-structure and data-value dependencies can affect the correct business process execution. However, if the data provided at runtime for the activities that conform the model have not got enough level of quality, then business process will not be successfully executed. When a BP is executed, it implies that the straightforward management of the data is exchanged between activities or that the data are acquired from external resources in order to compose the final outcome data. Consequently, the data that generate and compose the final product is considered critical, and is essential for the BP (Wang, 1998). Among other factors, it can be said that the success of an instance of the processes is grounded in the quality of the data used. The management of data with the adequate level of quality constitutes a key value for the successful execution of these processes. In order to make organizations aware of the importance of data-quality, a data-quality management plan should be implemented that covers the main data and data-quality requirements in the BP. The scope of both kinds of requirements should already have been dealt with by business experts. On the other hand, IT people should be able to implement the corresponding mechanism to satisfy the stated requirements. Furthermore, both business experts and IT people (along with Data Quality experts) have to decide how to incorporate the data-quality requirements through BPM life-cycle phases as suitable mechanisms and make them available for use. Once the data-quality requirements are gathered, the next step is to adapt the BPs to support the data-quality analysis without altering their fundamental structure and behaviour but taking into account the data-quality aspects.

Although certain data-quality related studies could be used at the design phase, such

as (Cappiello et al., 2013; Pipino et al., 2002; Rodríguez et al., 2012), there is a lack of proposals for their *system configuration* and *process enactment* phases: a necessity which we intend to cover in this contribution. Therefore, not only do we propose a theoretical solution, but we also define the steps to obtain an executable quality-aware BP using an imperative model with data management.

This chapter analyses the different modifications that a BP must suffer to be data-quality aware. In order to minimize the modifications over the model itself, and to facilitate both the model and the needed implementations to be able to deploy it, we propose the modification of the traditional PAIS. In (Dumas et al., 2005), the authors introduced the concept of Process-Aware Information System (PAIS) for facilitate the specification and enactment of business processes. One of the main advantages is that a PAIS separates process logic from application code (Weber et al., 2009). Specifically, in the PAIS framework, models of business processes are represented in the Process Layer, and the functionality of their activities are implemented and deployed by services situated in the Application Layer. We propose that certain data-quality management activities (e.g. measurement, assessment, or improvement) can be both supported by and implemented as part of the PAIS. The main consequence of our hypothesis is that part of the data-quality management activities, that should be implemented as part of the BP, can be now externalized as services. It makes possible that the BP, where required, can simply invoke these data-quality activities. Therefore, the modification of the traditional PAIS is proposed with the aim of supporting and addressing the data-quality management in various phases of the BPM life-cycle, facilitating the incorporation of data-quality aspects and reducing the complexity to implement them. In our proposal, all the data-quality activities in charge of the control and enhancement are externalized and gathered in a new Data Quality Layer, called the DQ Layer. To guide the incorporation and usage of this new DQ Layer, a methodology is also provided to help and support both business and data-quality experts through the various phases.

The remainder of the chapter is organized as follows: Section 6.2 proposes a case study which could be successfully applied to our problem. Section 6.3 presents the PAIS-DQ framework, extended with the DQ Layer to address DQM. Section 6.4 shows the steps to transform a BP into a data-quality aware BP. In Section 6.5, our proposal is applied to the case study. Section 6.6 includes an overview of relevant work. Finally, conclusions are drawn in Section 6.7.

6.2 Detailing a Case Study

In order to let the readers achieve a better understanding of the benefits of this proposal, at this point a brief summary is introduced of one of the case studies covered. This case study is an adaptation of the Trip Planner example presented in Section 2.6. This corresponds to the process only aimed at finding and booking the cheapest flight to make a trip according to customer requirements. Increasingly, people use marketplace applications on the web that integrate several results from queries to a variety of flight providers. A possible BP using the PAIS framework is shown in Figure 6.1. Business Process Model and Notation (BPMN) (OMG, 2011b) is used to describe the different

activities that constitute the BP, namely:

- (i) Firstly, the customer, through the presentation layer, introduces his or her travel requirements.
- (ii) Several activities are then executed in parallel to search for the flight that best meets customer requirements. These activities invoke a number of external services to obtain the flight information.
- (iii) The best flight is chosen, typically according to the cheapest price.
- (iv) The customer is informed of this flight.

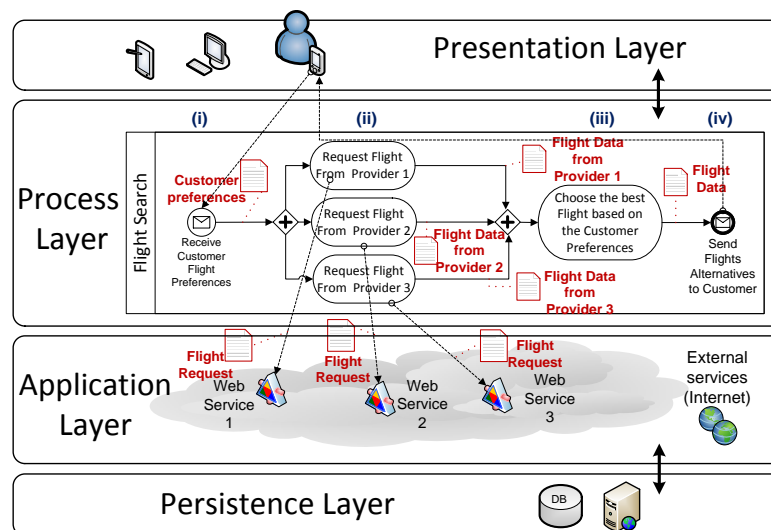


Figure 6.1 – Illustrative Example: Flight Search process.

In particular, the process model described in the Process Layer includes:

- The activities (“*Request Flight From Provider 1*”, “*Request Flight From Provider 2*”, ...);
- The data generated that flow through the process and is exchanged with the Application Layer (drawn in red, “*Customer preferences*”, “*Flight Request*”,...);
- And the control-flow (*AND* gateway in the example).

At the same time, in the Application Layer, the external services invoked by the activities are deployed and ready to be called (“*Web Service 1*”, “*Web Service 2*”, ...).

In each step of the process, a set of data is created, queried and updated. Special attention should be paid to the data obtained from external resources (step (ii) of the

example), and to the decisions taken based on the just-received data (step (iii) of the example). In other words, it is paramount to observe the levels of quality of the data of interest in steps (ii) and (iii). In our case, data-quality concerns are addressed for the set of data used by the activities. In a particular case, the data whose levels of quality should be borne in mind due to its importance, are on the one hand the *customer preferences*, which could be represented as presented in Section 3.2.1 by *departingFrom*, *goingTo*, *departDate* and *returnDate*; and on the other hand, the *flightInformation* could be represented by *flightNumber*, *carrier*, *departureTime*, *arrivalTime*, *priceFlight*, *checkedLuggagePrice* and *creditCardCharge*. After introducing our proposal, we will return to this case study to describe the steps taken to ensure the quality of this specific data.

6.3 PAIS-DQ Description

Based on the definitions given in 2.5.1, let us take into consideration the following points as the main rationale for our investigation: (i) the idea of considering data as a product (Wang, 1998), which enables the application of basic quality principles to the data; (ii) the need to consider only the quality of data in the process design phase; and (iii) the increment of the complexity of the model to include quality and more control-flow tasks.

We propose PAIS-DQ as a way to simplify the execution of some of the low-level DQ management activities. These activities articulated as part of the high-level DQ management activities are defined by the organization in order to ensure the overall level of quality of the data used in their running BPs. The simplification includes the possibility of externalizing the services to enable their reusability at different points of the BP. Therefore, the objective is the attainment of advantages of having externalized, to any BPs, the set of reusable mechanisms that are aligned to DQ strategy.

The set of low-level activities includes measurement, assessment, and enhancement of data; these activities should be considered as atomic operations for the high-level DQ management activities of DQ control and/or DQ assurance. In other words, if an organization is willing to grant its BPs DQ awareness at design level (Cappiello et al., 2013), then the PAIS-DQ offers the means to do so at an implementation level.

This section describes how the PAIS-DQ is structured. Along with the extension of PAIS, we also considered that, in order to support business and DQ experts and to enable PAIS-DQ to be used, some methodological guidelines must be provided: PAIS-DQ-HOW.

6.3.1 PAIS-DQ Architecture

We propose extending the classic PAIS framework (Weber et al., 2009) with the so-called DQ Layer, at the same level as the Presentation, Process, and Application Layers (see Figure 6.2).

BP models designed in the Process Layer establish communication with the customer through the Presentation Layer and run the services implemented in the Application Layer, since it is possible to combine several services within the same process. The DQ Layer is in charge of providing, in an external and independent way, the necessary

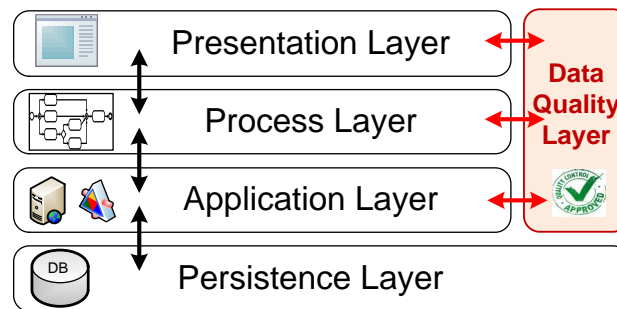


Figure 6.2 – Framework for Data Quality Management.

functionalities and mechanisms to manage the level of quality of the data that flow through the process in each instance. To this end, not only the Process Layer is affected by the DQ Layer. Firstly, certain mechanisms must be provided to enable the customer to establish the required DQ level. These mechanisms must be provided by the DQ Layer at the Presentation Layer level. At the same time, in the process model of the BP under study, it must be specified which data are to be controlled and/or assured.

Therefore, the DQ Layer must provide the necessary mechanisms to the Process Layer to locate, specify and define the data involved in the low-level DQ management activities. And finally, the services in charge of the DQ functionalities (i.e. activities) are implemented, deployed and located by the DQ Layer in the Application Layer.

Table 6.1 summarizes the various needs in each of the four PAIS Layers: DQ Layer Necessities, What needs to work; DQ Capabilities, What is provided; and DQ Layer Responsible, Who is in charge of configuring these capabilities.

Table 6.1 – DQ Layer in PAIS-DQ

PAIS Layer	DQ Layer Necessities	DQ Layer Capabilities	DQ Layer Responsible
Presentation Layer	Required DQ level by customer	Data defining required DQ level	Business and DQ Expert
Process Layer	BP model + DQ level + required Low-level DQ management activity	BPMN enriched with DQ requirements	Business and DQ Expert
Application Layer	DQ Services	DQ functionality	DQ Expert
Persistence Layer	-	-	Business Expert

On the other hand, once data used in the BP is analysed and/or modified in the DQ Layer, the information about the level of DQ should be included in the BP model. For this reason, a new type of data is included in order to make the BP data-quality aware. This new data type is called *Data-Quality Items* (DQ Items).

Hence, *DQ Items* constitutes the set of data that contain the information about the level of quality of a specific item of data and/or a specific dataset, and is related to a set of DQ dimensions. Therefore, these DQ Items should point to other types of data, since they could include additional information related to the level of quality. In our case study, the piece of data called *arrivalTime* can be enriched with the *DQ item* corresponding to its *accuracy* information (e.g. level required, moment in the BP where it must be controlled and/or assured, and value obtained). How this information should be included in the data-flow is detailed in Section 6.4.

6.3.2 Data Quality Management Activities

As explained earlier in Section 2.5.1, the high-level DQ management activities (control and/or assurance) are to use some low-level DQ management activities (measurement, assessment, decision-taking, enhancement). The relationship between these activities is shown in Table 2.3. How details concerning these DQ management activities can affect the BP, and how PAIS-DQ support this influence are introduced in the following subsections.

6.3.2.1 How High-Level DQ Management can affect the BP Model

When an organization decides to make its business processes “data-quality aware”, then they high-level DQ management activities (control and/or assurance) should be included into the BP. This decision implies modification of the process model.

Implementing DQ control involves inclusion of the low-level DQ management activities of measuring, assessing and/or enhancing data. To this end, business experts, together with the DQ experts, must decide whether or not the BP model should be changed, and if necessary, how to introduce these new activities. For example, a BP model should not be changed if the consequences of including decisions related to DQ only imply the modification of the conditions associated to the branches of the flow, or if the consequences only affect one activity. Specifically, Figure 6.3 shows how the conditions to take the different outgoing branches of an exclusive gateway are enriched with the assessment results.

On the other hand, on implementing DQ assurance in order to ensure a specific level of DQ in a BP, both types of expert should study and evaluate how, at a certain point in the process, a set of DQ requirements must be implemented. For example, a level of DQ can be assured by enhancing a set of data, or possibly, by going back to a point at which the level of DQ was acceptable, such as shown in Figure 6.4, where if the DQ level of the flight data from the provider is assessed as “not good”, then the provider is asked for flight information again.

6.3.2.2 How Low-Level DQ Management Activities can affect the BP Model

The implementation of low-level DQ management activities brings several consequences to the BPs:

- **Leverage of Internal Performance:** This strategy proposes the implementation of some of the previously stated low-level DQ management activities inside the

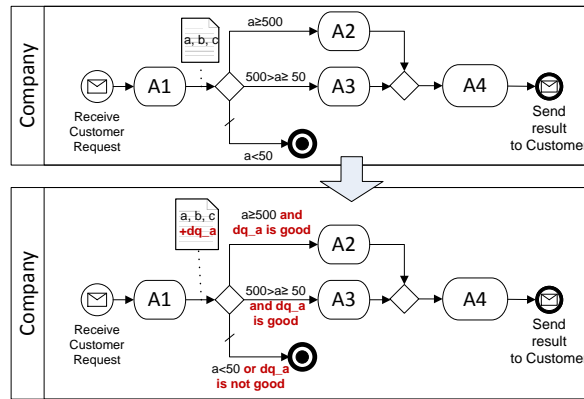


Figure 6.3 – Data Quality Control by including new decision rules in a gateway.

BPs. This commonly implies the inclusion of new activities within the BP model by using some of the programmable languages that exist in a Business Process Management System (BPMS). In (Caro et al., 2012), the authors show how these types of activities can be included in the BP as new activities. Specifically, a new activity is included for the overall assurance of the level of quality for a DQ dimension required.

- **Leverage of External Performance:** External services are responsible for providing the low-level DQ management functionality. Therefore, the activities that need to measure, assess and/or enhance the level of quality of some data must call the external service that contains the necessary DQ functionalities. This external enactment can be invoked from:
 - One of the existing activities in the BP, such as those shown in Figure 6.5.
 - A new activity created for the invocation, as proposed in (Cappiello et al., 2013), (Caro et al., 2012) and (Rodríguez et al., 2012).

In this work, externalization into the DQ Layer is proposed of as many of the low-level DQ management activities as possible, as shown in Figure 6.6.

6.3.3 Data Quality Layer Functionalities

In our proposal, we strive to optimize the implementation of the DQ management activities from the point of view of the development of the software systems that support the BP. It is assumed that the DQ Layer can minimize the degree of modification required of the BP process model and the corresponding software system(s) that support the BP. Consequently, the more these functionalities can be externalized, the less the software system(s) will need to be modified. It is therefore intended, by means of the DQ Layer, to enable the reduction of the implementation time for the specific DQ requirement for each problem. The various activities of the BP process must interchange the *DQ items*

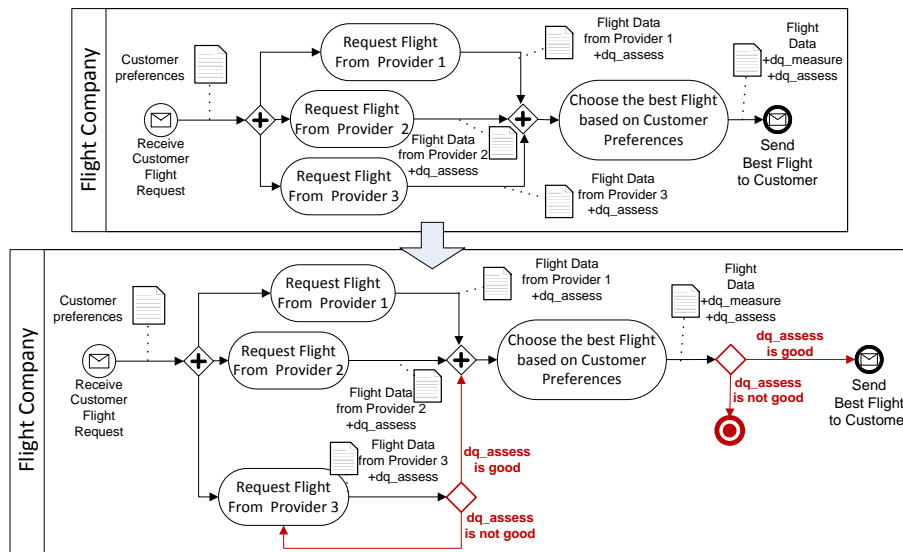


Figure 6.4 – Data Quality Assurance by inclusion of new branches.

with the corresponding activities of the DQ Layer associated with measurement, assessment, and/or enhancement. Consequently, the functionalities to be included as part of DQ Layer are determined by the low-level DQ management activities required to render the process DQ aware, such as DQ Measurement, DQ Assessment, and DQ Enhancement (cf. Section 6.3.2).

The deployment and usage of the various functionalities could be performed by taking into account the technologies that best fit the BPs. For example, web-service technology has been demonstrated to be highly useful (van der Aalst et al., 2003).

6.4 PAIS-DQ-HOW: Methodology to use PAIS-DQ

A methodology is proposed in order to ease the utilization of PAIS-DQ and to provide some structured guides that enable IT people to use the corresponding DQ Layer. Compared to certain proposals in the literature, such as BPiDQ by (Cappiello et al., 2013), our proposed methodology, called PAIS-DQ-HOW, goes beyond the design phases of BPs. In fact, not only does it cover the Design Process phase of the PAIS life-cycle, but it also completes the rest of PAIS life-cycle phases to facilitate the inclusion of DQ management activities into the BP.

In addition, PAIS-DQ-HOW should cover:

- The design of the high-level DQ management activities in the organization.
- The design of the low-level DQ management activities.
- A full description of the DQ items that should be exchanged.
- The design and deployment of the DQ Layer.

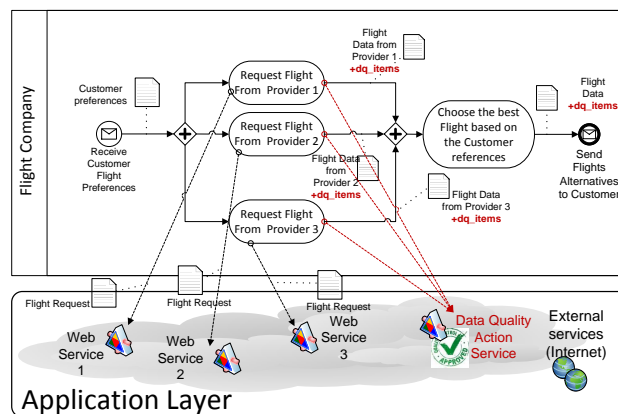


Figure 6.5 – Data Quality Action Performance in an external service.

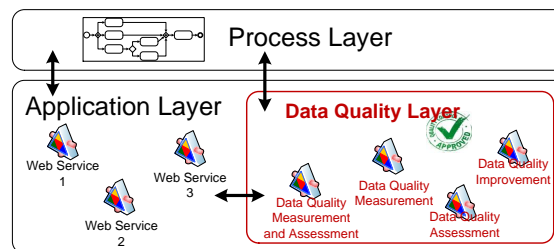


Figure 6.6 – Data Quality Layer services.

- How low-level DQ activities are included in the BPs by using the DQ Layer to achieve the goals of the high-level DQ activities.

Therefore, how these inclusions affect the various PAIS-DQ Layers is described and specified through the PAIS life-cycle phases.

6.4.1 Business Process Design

BP models contain all the information about the activities, the data-flow, and the sequence-flow located in the Process Layer. In addition, the necessary services in charge of the functionality of several activities are situated and described in the Application Layer. BPs can be modelled in any Business Process Management System, or can also form the input of the methodology. This step can be skipped if the BP already exists and if the objective is to make this BP DQ-aware.

6.4.2 Data Quality Layer and System Configuration

Once the process is modelled in the Process Design phase, the business expert together with the DQ expert must configure the process (System Configuration phase) and decide

which DQ management activities must be considered in the BP. Both experts must then also decide the data affected, the BP changes, and the specific implementation.

6.4.2.1 Configuration of High-Level DQ Management Activities

The organization, taking into account its specific needs, should identify which activities (control and/or assurance) should be implemented to obtain greater benefits. This also implies the selection of those parts of the BP are susceptible to requiring special attention for the data used.

As part of the control and/or assurance, not only does data requiring control and/or assurance have to be identified, but also the DQ dimensions representing the criteria that should be covered. Developers should therefore specify the DQ requirements for each item of data through the specification of the following information:

- **Data:** the data that should be controlled and/or assured.
- **DQ Requirements:** the DQ characteristics that are going to be considered, (e.g. completeness, accuracy, and credibility). These DQ characteristics, along with their corresponding values, are included as new DQ items into the data-flow.
- **Who measures:** who is in charge of the measurement of the level of DQ.
- **Who assesses:** who is in charge of the assessment of the level of DQ.
- **Who enhances:** who is in charge of the enhancement (improvement) of the level of DQ.

As a result, several *DQ items* are identified and must be fully described. Furthermore, their usage has to be standardized across the BP.

In order to communicate to the final customers the state of the data that they are about to use, these DQ items are exchanged between the layers of the PAIS. The way to calculate the value for these DQ items in the BP is performed through the corresponding low-level DQ management activities. These activities should establish various algorithms and mechanisms for the calculation of the value for each DQ item as explained below.

6.4.2.2 Configuration of Low-Level DQ Management Activities

In order to conduct the design of the high-level DQ management activities, organizations should coordinate the required low-level DQ management activities.

There are several possible configurations for the low-level DQ management activities. Therefore, along with the identification of the activities, in this stage, certain further elements must be defined:

1. The set of low-level DQ management activities that are part of the high-level DQ management activities.

2. The interface for each activity. As part of this interface, the corresponding input and output data (parameters) that enable customization to different scenarios should also be defined. Examples of these parameters include the data requirements related to certain DQ dimensions, such as the measurement related to completeness, or the assessment related to completeness.
3. The corresponding algorithm/mechanism for the measurement method, assessment method and/or enhancement methods. For example: for measurement, the corresponding measurement methods (the algorithms) have to be defined; for enhancement, the corresponding enhancement of data (including possible sources) should be identified.

Once the low-level DQ management activities are designed, the corresponding implementation must be developed and deployed. In this case, the DQ expert, in representation of the entity (organization or department) in charge of performing the DQ aspects, must develop the necessary implementation in order to make these low-level DQ management activities available.

6.4.2.3 Changes to the BP to make it DQ aware

Therefore, depending on the high-level DQ management activities, the set of changes to apply to the BP varies. On the one hand, the data-flow is always modified to include DQ items. On the other hand, the experts should act accordingly on the control and/or assurance necessities, as explained previously, (e.g. externalizing the measurement, including a decision rule in a gateway, and including a new activity to improve a set of data). In other words, the adaptation or redesign of the BP includes the low-level DQ management activities that form the high-level DQ management activities planned by the organization.

6.4.3 Business Process Execution

Once all the parts of the extended model have been defined, the executable process can be performed. When an instance reaches an activity connected with an external service in charge of the DQ aspect, then the values for the data of the instance and the requirements and thresholds (if necessary) are sent to this service. Otherwise, when an instance reaches an activity in charge of the DQ aspects, then the values for the data, the requirements and the thresholds (if necessary) are taken from the incoming sequence-flow.

The synopsis of which PAIS-DQ Layer is affected in each step of the methodology, what is used, what is obtained, and who is responsible is detailed in Table 6.2 below:

6.5 PAIS-DQ Applied to the Case Study

In order to illustrate the usage of both the methodology and the DQ-PAIS, in this section an explanation is given as to how to render the case study DQ-aware. To this end, the

Table 6.2 – PAIS-DQ-HOW Methodology

Methodology Stage	PAIS-DQ Layer	What is used	What is obtained	Responsible	
1. Process Design	PAIS	PAIS	BP Model	Business Expert	
2. DQ Layer and Sys.	PAIS	PAIS-DQ	BP Model + DQ	Business, IT and DQ Expert	
2.1. High Level.	Presentation and Layers	Process and Application Layers	BP Model	High Level Activities + DQ Items	Business and DQ Expert
2.2. Low Level.	Application Layers	High Level Activities	BP Model + DQ	Low Level Activities and Implementation	Design and DQ Expert
2.3. Changes to BP.	Process Layer	BP Model + DQ Requirements	BP Model with DQ	Business and IT Expert	
3. BP Execution	PAIS-DQ	PAIS-DQ	Executable BP	IT Expert	

three steps of our aforementioned methodology are applied and detailed in the following subsections.

6.5.1 BP Design: Flight Search Process

The BP used here is that presented in Section 6.2 as a case study. This process can be modelled in any Business Process Management System, such as IntalioTM (Community, 2012b), ActivitiTM (Team, 2012), and Bonita Open SolutionTM (Community, 2012a). In our case, Bonita Open SolutionTM is applied to design and execute the BPs, as shown in Figure 6.7, since it is an open-code application with a free distribution, and is commonly used in the private market.

6.5.2 DQ Layer and System Configuration

In our case study, we are specifically interested in implementing a **DQ control** plan, since we consider that customers can be aware of the level of quality of the data they are using when booking flights. Therefore, the DQ Layer and the System configuration is focused on providing and developing the necessary mechanisms to control the level of DQ in the Flight Search Process. Therefore, the aim of this stage is to make a BP become DQ-aware through the configuration of its DQ items, required low-level DQ activities and implementation of these low-level DQ activities.

6.5.2.1 Configuration to Control Data Quality Levels

Regarding the data that are required to be controlled, we focus on that corresponding to the outcome data of the activities *Request Flight from Provider 1*, *Request Flight from Provider 2* and *Request Flight from Provider 3*. These items of data in the case study are the result of the data exchange between these three activities and the related external services. Table 6.3 provides details of the information needed, given by the DQ Experts, for some of these items of data. In this case study, no one enhances since it is not required, therefore, this information is omitted.

Table 6.3 – Data Quality Requirements for *Request Flight from Provider 1* activity output

Data	Dimension	Level	Control/ Assurance	Who measures	Who evaluates
arrivalTime	Accuracy	High	Control	I8K	I8K
checkedLuggagePrice	Accuracy	High	Control	I8K	I8K
creditCardCharge	Completeness	High	Control	I8K	I8K
...

6.5.2.2 Data Quality Items

The DQ dimensions chosen were **completeness** and **accuracy**. For these DQ dimensions, it was decided to include this information into that which should be communicated to the

customer. This information is given by the DQ items. These DQ items must be managed as part of the operation of the DQ Layer, and must be exchanged from and to the DQ Layer and the corresponding activities in our example. See the next subsection for a more in-depth description.

6.5.2.3 Low-Level DQ Activities to Control DQ Configuration

On the other hand, since we have chosen a DQ control plan, and taking into account the set of activities in Table 2.3, the low-level DQ management activities that should be implemented and deployed to the DQ Layer are the measurement, assessment, and decision on what to do. Therefore, an explanation is given below on how these DQ requirements are carried out through a BPMS.

As part of this process, details are shown in Figure 6.7 on how to measure the level of quality of the data produced by the activity *Request Flight from Provider 1* (first step of the business process).

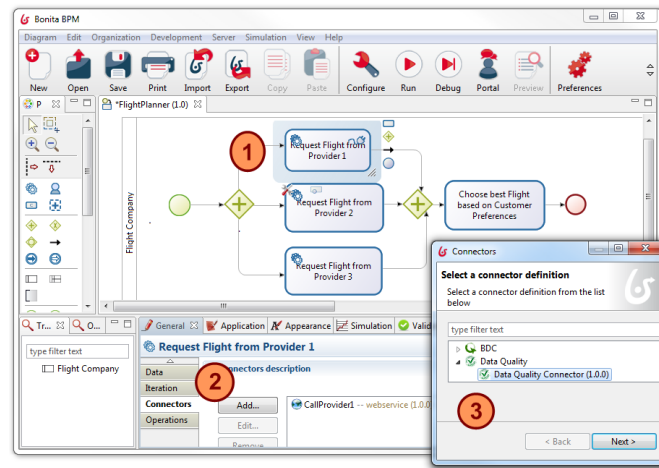


Figure 6.7 – Add a *Data Quality Connector* to an activity with Bonita Open SolutionTM.

Since the DQ measurement is externalized and located in the DQ Layer, one way to measure the level of quality of the data returned by a provider involves the addition of a *Data Quality Connector* to this activity (second step). A *connector* is the way in which Bonita Open SolutionTM links an activity with the service or application that executes a functionality, such as the DQ measurement.

The connector, called the *Data Quality Connector*, has been designed with the aim of specifying the DQ requirements in order to send the data values at run-time to the software in charge of the DQ measurement, assessment, and/or improvement. In other words, it connects activities in the Process Layer to the DQ Layer to obtain the results of DQ functionalities. In addition, the *Data Quality Connector* also permits connector outputs to be retrieved and to store them in the process variables. As shown in Figure 6.8, the DQ requirements detailed in Table 6.3 are indicated through the wizard provided and then the result obtained by the service can be stored in the DQ items of the BP. In the example, the DQ items are: *arrivalFlightProvider1Accuracy* and *creditFlightProvider1Completeness*.

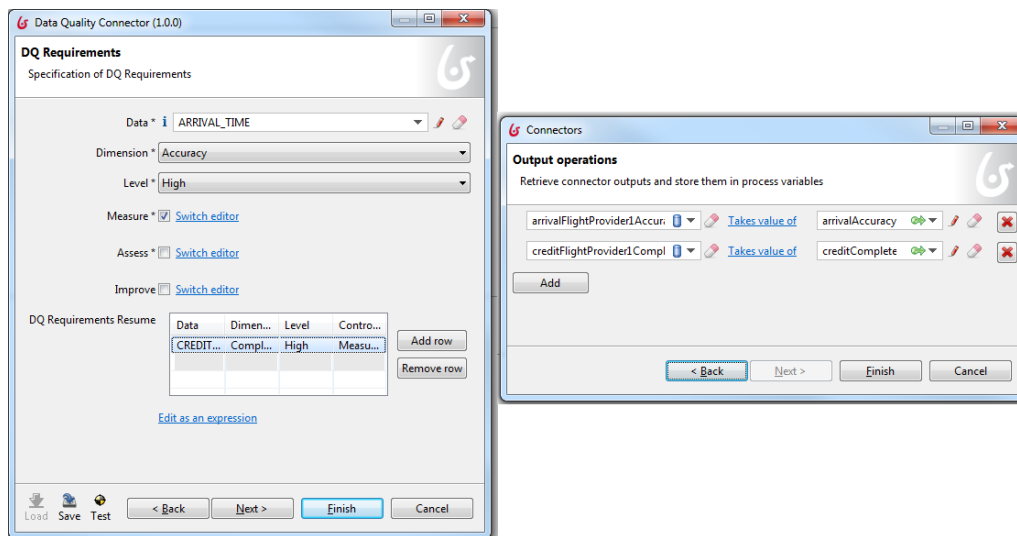


Figure 6.8 – *Data Quality Connector* configuration for the activity *Request Flight From Provider 1*.

6.5.2.4 Low-Level DQ Activities to DQ Control Implementation

It is possible to find various solutions that implement the necessary functionalities to support the operation of the DQ Layer. The implementation must support several combinations of the aspects described in the previous section. In this section, the opportunity of adding certain information concerning the certification of DQ levels is also considered in order to increment the reliability of the data. This information is to be included into the DQ items. Once the information of the certification of the data is conveniently supplied, the process can be adapted so that it may be managed, and decisions, based on such information (control and/or assurance), may be made in execution time.

Specifically, we have developed a service architecture, named I8K and previously introduced in (Caballero et al., 2013), which satisfies the requirements for the DQ certification schema given by the ISO 8000-1x0:2009 family (ISO, 2011a). These requirements consist in the incorporation of certain information on the data being exchanged between the Process and the Application Layer. As part of this information, the certification of the DQ is included (see Figure 6.9).

```
<completeness-event event-type="certify140" organization-ref="
  I8K" date="2013-05-21T23:49:58.213+02:00">100 %</completeness
-event>
```

Figure 6.9 – Fragment corresponding to the completeness dimension given by I8K Architecture.

More specifically, the standard supports the certification of only those two DQ dimensions chosen: accuracy in ISO 8000-130 (ISO, 2011b), and completeness in ISO 8000-140 (ISO, 2011c). To the best of our knowledge, there is currently only one public usable implementations of standard ISO 8000-1x0:2009: that developed by ECCMA, which is available

at (Benson and Hildebrand, 2012b) under payment. However, the I8K implementation strives to satisfy all of the requirements established in the various parts of the family of standards. This motivated our decision to carry out our own implementation.

This I8K therefore provides our DQ Layer and supports the low-level DQ management activities needed for our case study.

6.5.2.5 *Flight Search Process Changes*

While focusing on the case study, let us explain one of the changes to be made: as part of the DQ control, the business and DQ experts must adapt the *Choose the best Flight based on Customer Preferences* activity so that they are responsible for verifying and deciding how the decision can affect these DQ levels. For example, if a customer establishes, as a DQ requirement that the landing time (arrivalTime) has to be accurate, then it is expected that if any of the activities returns a flight with a landing time without specifying whether it is “a.m.” or “p.m.”, then, this flight is not considered by the activity *Choose the best Flight based on Customer Preferences*, and therefore, not offered to the customer.

On the other hand, for the activity *Request Flight From Provider 1*, it was decided to include the measurement of the DQ. To implement the measurement, two connectors to the activity were required: the first, which connects with the provider; and the second, which connects with the entity in charge of the DQ measurement, such as shown in Figure 6.10. This implementation was in a similar way for the remainder of the activities.

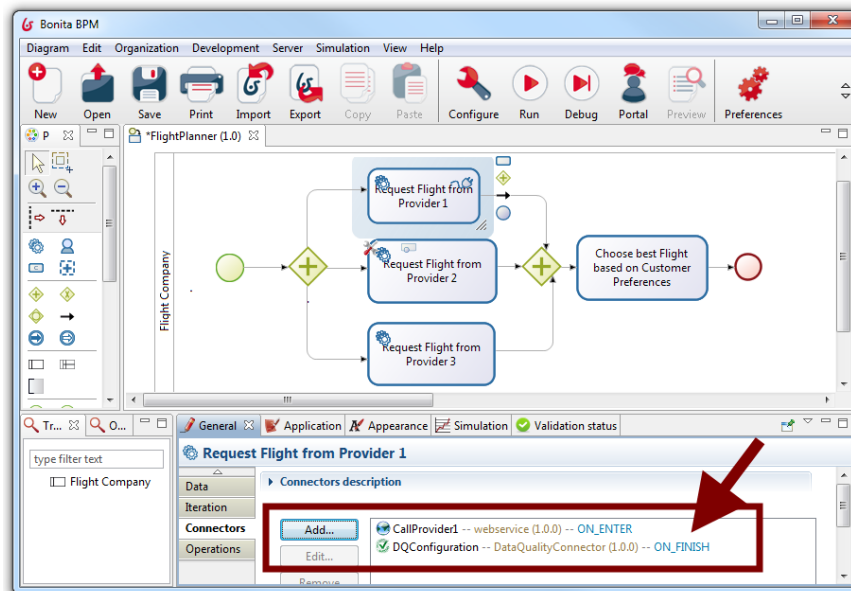


Figure 6.10 – Connectors in Bonita Open SolutionTM for the activity *Request Flight From Provider 1*.

6.5.3 BP Execution

Once all the parts of the BP have been defined, the execution process can be performed using the execution engine of Bonita Open SolutionTM. The selection of the best flight now takes into account the DQ items containing information about the completeness and the accuracy, and that information can be used accordingly.

6.6 Related work

Once the underlying concepts about PAIS and DQM have been studied, our purpose is to include DQM activities into PAIS in a transversal way. This enables BP to take advantages of using the various activities of DQ management. This implies the capacity to ascertain the level of quality of the data that flow through a process without devaluating the process itself. It means that it would be possible to implement the DQ management activities, along with any activities or decision processes, without the need for implementing a specific solution to manage the DQ in each case. Therefore, an increase in the number of the layers is implied (Presentation, Process, Application and Persistence) with any other layers that have specific goals. In (Jablonski and Bussler, 1996), Jablonski and Bussler identified other important perspectives for PAIS: causality, integrity and failure recovery, history, security and quality. The quality perspective is related to the “*establishment of a control mechanism to determine whether a process instance has been executed in an efficient manner or not*”. However, they fail to define a DQ perspective related to the level of quality that data should reach. In (Gómez-López and Martínez Gasca, 2010a) and (Gómez-López and Martínez Gasca, 2010b), Gómez-López et al. present an extension of the PAIS framework, where an analysis of the correctness of the data stored in the Persistence Layer is proposed in order to diagnose the incorrect data according to the Business Rules of the process, but this extension is not related to the DQ aspects. Furthermore, related to DQ aspects, there is otherwise a wide variety of applicable works. In (Marotta et al., 2012), Marotta et al. highlight the importance of applying data-quality in Web Warehouse (WW). They present a framework to include DQM in WW life-cycle with the aim of develop a data-quality aware WW. This awareness is obtained by means of filtering undesirable items and also to guide the processes of source selection and extraction.

On the other hand, there are various studies that specifically focus on how to design quality-aware BPs. In (Heravizadeh et al., 2009), Heravizadeh et al. identify which DQ dimensions should be analysed in a BP. Cappiello and Pernici, in (Cappiello and Pernici, 2006), describe a methodology for integrating some concerns related to DQ management, specifically, on how BPs should react when errors due to poor DQ occurs during the enhancement of the Web Services. However, they focused their research on the detection and correction of errors of data exchanged by the services and found at runtime. In addition, they also analysed the correct way of working for an activity based on this data. The main difference to our work is that we study how to better address the various ways to measure, assess, control, improve and assure the DQ level in a BP that is supported by a PAIS, and not just the possible errors of these data. On their part, Rodríguez et al., in (Rodríguez et al., 2012), propose a BPMN extension to model several DQ

aspects. Nevertheless, their focus is on a descriptive approach rather than an analytical approach. Following on from that work, Caro et al. (Caro et al., 2012), and Cappiello et al. (Cappiello et al., 2013) tackle the problem of how the BP is affected by the management of DQ by defining a methodology called BPiDQ to consider DQ issues in the BP modelling phase; however, they fail to consider how this BP is affected at runtime when the model is executed in a BPMS.

Furthermore, after conducting a systematic literature review, it is found that none of the proposals encountered specifically address this DQ perspective in the PAIS Framework and through any of the PAIS life-cycle stages. In addition, we demonstrate this proposal with an implementation of these ideas in a BPMS.

6.7 Summary and Discussion

This contribution proposes a methodology to manage the data-quality in business process, a key characteristic in data management. In order to do applicable this methodology, we propose an extension of the Process-Aware Information System framework, called PAIS-DQ, with the aim of including data-quality aspects into business processes. The control and/or assurance of data-quality should not be given by the activities that shape the process, since the level of data-quality has to be guaranteed by an external and independent entity. PAIS-DQ includes a Data Quality Layer in charge of incorporating these quality aspects into the BP, thereby avoiding modification of the BP model itself with data-quality aspects, and maintaining a separation between the BP model and how the quality level is obtained. Thanks to this layer, it is possible to execute a BP instance which not only covers the preferences of customers, but also surpasses their expectations with regards to data-quality. Data quality awareness is incorporated by means of the inclusion of these data-quality aspects into a BP model using DQ Layer.

The usage of the methodology and the DQ Layer have been illustrated by applying them to a case study, in which we have shown how to introduce several data-quality activities to control the levels of completeness and accuracy of the data. In our case study, Bonita Open SolutionTM is applied to design and execute the BPs. Specifically, a *Data Quality Connector* has been designed within Bonita Open SolutionTM with the aim of specifying the DQ requirements in order to send the data values at run-time to the software in charge of the DQ management activities. I8K, which meets the requirements of ISO 8000-100 to ISO 8000-140, has been used as the implementation of the DQ Layer, to provide the support to the low-level data-quality activities.

Part VI

Conclusions and Future Work

Chapter 7

Final Remarks

Your work is going to fill a large part of your life, and the only way to be truly satisfied is to do what you believe is great work. Don't settle.

Steve Jobs.

The current Thesis Dissertation presents three main contributions to support the data-oriented optimization in business processes. These contributions have been orchestrated under the umbrella of Combi-BP Framework to improve the data aspect in Business Processes. The framework presents the process of specifying a data-oriented optimization problem in a declarative way, and how it is transformed into an imperative model taking into account the data-structure and the data-value dependencies. This imperative process maintains the flexibility of the declarative specification, thanks to the use of constraint programming, but enables its deployed and is closer to an executable model deployed in a commercial BPMS. In order to cover various data aspect, how the data-quality dimensions can be included in a business process is also dealt.

In order to shield the business experts to unnecessary details and to help them to describe the a data-oriented optimization problems in business process in a more flexible and adaptable way, we propose a declarative language where the data aspects are included. This declarative language allows the business experts to describe what they want instead of how to obtain it. This declarative model will be transformed into an imperative model. The business experts will not have to think about how to design the BP where the activities need coordination to obtain a concrete combination of data, only to specify the BP requirements, leaving the analysis of the best activities modelling and the model to the framework.

- Part III formalizes the type of problems dealt in this Thesis Dissertation. We define a data-oriented optimization problem in business process as the set of activities that are related and have to optimize the outcome of the process. This optimization means that the order of the activities and the final product depends on the values of the data handle in each instance. This problem implies a depth study and analysis

of how to design an executable BP capable of check every possible combination of data to obtain the optimal outcome. In order to describe a data-oriented optimization problem, two proposals are presented. Firstly, a Data-Oriented Optimization Language, called DOODLE, is developed. DOODLE provides a graphical notation to describe the set of activities, their input and output data, and pre- and post-conditions, the set of constraints that relate the activities, and objective function. Secondly, an extension of the standard BPMN has been described. This extension enables to include the declarative description of the BP as a new type of sub-process in a BPMN model, allowing the combination of imperative and declarative descriptions in the same model.

These contributions have been addressed in various publications. Related to DOODLE the following publications have been published: (Fernández Narváez, 2013; Parody et al., 2013a,b). Furthermore, an article including DOODLE has been submitted as to an indexed journal (Parody et al., 2014c). Related to BPMN extension, an article in the International Workshop on the Business Process Model and Notation focused on the latest developments around BPMN (Parody et al., 2012a). In addition, a tool has been developed (Parody et al., 2013c) and (Parody, 2013) to support the data-oriented optimization description within an imperative BP model using BPMN. This tool has been also registered in the Intellectual Property Record of Andalusia (Spain).

However, this specification is insufficient to be deployed neither executed in a commercial BPMS. This incapacity brings us to develop a set of configurations and transformations to convert this declarative model into an imperative model. The configuration permits to establish an order between the activities in function of data-structure dependencies. This order depends on the data constraints and the relationships between the activities, but also remains insufficient to support the data-value dependencies to find the most appropriate input data to optimize the outcome data. In order to solve this problem, two transformations have been developed to solve locally, or by executing the activities, but guaranteeing that the optimal outcome is found.

- Part IV presents a configuration based on the data-structure dependencies, and two approaches to solve the data-values dependencies, as detailed as follows:
 - Chapter 4 is focused on the configuration process. The configuration enables to transform the declarative model, formalized before, into an imperative representation. An automatic method based on Constraint Programming, and a set of developed algorithms, establishes an order between the activities taking into account the data relationship between the activities. This order is translated into an imperative model represented using the standard BPMN 2.0 (OMG, 2011b). Furthermore, the BPMN model obtained is optimal in the sense that the execution time of the instances is the minimum.
 - Two approaches are proposed in Chapter 5 in order to solve the data-value dependencies of data-oriented optimization problems in BP. Each approach

depends on the knowledge related to the functionality of the activities. If the values of the output variables of an activity can be ascertained through their pre- and post-conditions (white-box approach), then, the optimization problem can be solved locally by using Constraint Programming. Otherwise, if the activities have to be executed to ascertain the output values (black-box approach), then, an algorithm is developed to guarantee that every possible combination of data is checked, and the optimal solution is obtained.

These contributions have been addressed in various publications. Related to the configuration system, an article has been submitted to an indexed journal (Parody et al., 2014b). Related to white-box, the paper (Parody et al., 2013b) has been published, and an article has been submitted to an indexed journal (Parody et al., 2014c). The black-box approach has been addressed in the following publications (Parody et al., 2010, 2011a,b, 2012b).

One of the main problems to focus a business process to the final product optimization, is that as soon as the final product satisfy customer requirements, the successful of the BP will be greater. However, there are issues that are out of the company reach. For example, in the majority of cases, the data provided by the activities come from external entities. This implies that the final product will depend, in part, on the level of quality provided by these entities. This problem can be solved including a set of data-quality requirements in the BP in order to warranted that the final product has a high level of data-quality. One of the main disadvantages of including data-quality requirements in a BP is the besmirched of the BP.

- Part V presents an extension of the traditional process-aware information system and a methodology to include data-quality requirements in a business process. This extension enables to obtain the data-quality management advantages without modify the BP itself by externalized every data-quality functionality. Furthermore, this extension permits to execute a BP instance which not only covers the preferences of customers, but also surpasses their expectations with regards to data-quality.

This contribution has been addressed in various publications. (Bermejo, 2013; Bermejo et al., 2013; Caballero et al., 2013). In addition, two articles have been submitted to indexed journals (Bermejo et al., 2014; Parody et al., 2014a). Furthermore, a tool has been developed (Bermejo and Parody, 2013) to support the family of standards ISO 8000-1x0:2009, and has been registered in the Intellectual Property Record of Ciudad Real (Castilla La-Mancha, Spain).

In short, this Thesis Dissertation provides a complete support to data-oriented in BP. This support enables business expert to describe the problem composed by a set of activities, data, constraints, and an objective function to be optimized, and let to the different proposals, the transformation into an optimal imperative BP that obtain the optimal outcome, following the data-quality requirements needed in each case.

Chapter 8

Directions of Future Work

"The consequences of our actions are so complicated, so diverse, that predicting the future is a very difficult business indeed."

Albus Dumbledore.

In the current Thesis Dissertation, a significant research works are presented. All the proposed approaches can be extended with new research topics to be addressed in future work.

In the first instance, regarding the declarative specification centred on data-oriented optimization problems, the contribution provided in Part III is intended to be extended with the next ideas:

- The proposals provided in this thesis dissertation are oriented towards the optimization of an objective function. However, it would be interesting to study the inclusion of several objective functions, or multi-objective functions in the BP. For example, the customer could want the cheapest travel, but also the fastest one. A combination of both objectives enlarges the possibilities related to the optimal business product offered to the customers.
- The possible constraints that can be used in our proposal could limit the capacity of their expressiveness as allowed by the grammar and the type of variables. The constraints constitute the formal representation of the relations between the data that forms the BP. The limitations of use of the proposal appear when the constraints cannot be represented by the relations described; by means of the presented grammar; by the data type; or by the operators included in this proposal, such as when a relation between two variables is described by means of a trigonometric function. The limitation of the data domain and the operations that can be applied, are established by the solver for the Constraint Optimization Problem, explained in Section 3.2. Most of the commercial solvers maintain the capacity to include Float, Integer, Sets, Boolean, Date and String variables in the model, thereby making it possible to cover a significant number of problems and the constraints that they

need. For future work, we plan to extend the grammar of the constraint presented in this thesis dissertation to enlarge the expressiveness of the model.

- New types of data-oriented optimization processes could be analysed: for example, in the determination of whether it is necessary to execute certain activities apart from those which achieve the objective. Furthermore, the activities could be executed more than once for the optimization process in each instance. This means that the activities could be provide various results that compose the final business product.

The configuration problem presented in Chapter 4 by considering the data-structure dependencies, could be extended with the following characteristics:

- Since the specification of data-oriented optimization processes does not establish the execution time of each activities, the Algorithm 1 (cf. Subsection 4.3.1) creates a COP from the declarative specification assuming that every activities lasts the same interval of time: a unit of time t . Therefore, the real execution time of each activity could be specified in the declarative model and considered for the configuration proposal.
- The imperative model configured from the declarative specification includes start and creation of the BPMN model includes the following BPMN components: start and end events; tasks; and parallel, exclusive, and inclusive gateways. However, BPMN enables and defines more components in a BP. We propose to increase the capabilities of the configuration, such as by including loops within the model.

Related to the white-box and black-box proposals, the advances are related to the changes made to enlarge the capacities of the specification, since both proposals must be adapted to provide the functionality required when a multi-objective is required, there are new type of constraints, the activities can be executed more than once, and/or other BPMN elements are considered in the imperative process.

However, one of the main challenges for our future work is the application of our proposal to real scenarios. On the one hand, several real services to give the necessary functionalities to the activities that compose the trip planner example are being developed. The various services will take the data from existing providers on Internet. Unfortunately, few providers allow us to take data in order to test our proposal, so the negotiations to have the permissions are taking time. On the other hand, an application example, in which we are working on, is the smart cities process developed in Trento, Italy (<http://www.smartcampuslab.it/>). The main idea of the process is to improve travel experience for citizens/tourists in the Trentino area providing accurate, real-time, and customized mobility services supporting the whole travel duration.

- Provide more accurate and customized mobility solutions: possibility to provide specific solutions that take into account the user context and preferences (additional information for tourists, services for elderly people, disabled people)

- Support citizen awareness for smart mobility choices: possibility to combine different mobility services, present different alternatives and support the choice with relevant information (cost, duration, environmental impact, cultural/landscape attractions, users recommendations)
- Provide more travel options: not only traditional mobility services (parking, taxi, public transportations), but also less known/used services (car sharing/pooling, bike sharing)
- Support the whole travel duration: not only information services (availability, schedule, cost information) but also booking/payment services, and notification services (delays, cancellation, unavailability).

Our objective is to obtain the best route by taking into account the user context and preferences (additional information for tourists, services for elderly people, disabled people) and the run-time information (see Figure 8.1).

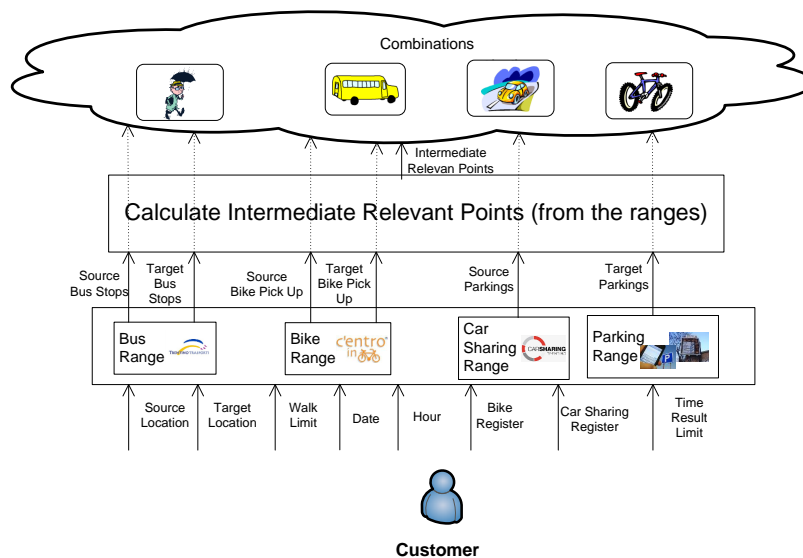


Figure 8.1 – Smart cities requirements.

A combination of several services related to various transports must be combined to establish a personalized route between two places in Trento city. This combination fits with the data-oriented optimization problems, but requires the possibility to execute the services more than once, since the customer can walk to go to a bus stop, and then walk again to go from the bus stop where the bus let the customer, and walk to his/her destination.

Related to the data-quality proposal explained in Part V, future work are related to the improvement of the I8K architecture by including more data-quality dimensions to the assessed and certified capabilities. In addition, the refinement of I8K by adding some

features, like the one regarding to security, would be advantageous in order to offer a more complete data-quality services. On the other hand, negotiations to conduct a pilot project with organizations from different domains that have different volumes of traffic data, could help us to tune up adequately the optimization of I8K architecture.

Finally, although each contribution has been implemented and tested independently, two of them even counting on their own registered tools, it would be desirable to develop a tool including all the functionality provided by Combi-BP framework. This way, the tool would support every approach explained in this thesis dissertation through a single access point.

Part VII
Appendices

Appendix A

Abbreviations

Throughout this thesis, we made use of abbreviations. Even though we carefully introduced abbreviations, for the sake of completeness and quick lookup, all abbreviations used in this thesis are listed in Table A.1

Table A.1 – Abbreviations

Abbreviation	Full Name
ADI	Activity Data Input
ADO	Activity Data Output
BP	Business Process
BPEL	Business Process Execution Language
BPM	Business Process Management
BPMN	Business Process Model and Notation
BPMS	Business Process Management System
Combi-BP	Combination of Activities in Business Process Framework
CP	Constraint Programming
CPN	Coloured Petri Net
COP	Constraint Optimization Problem
CSP	Constraint Satisfaction Problem
DF	Data-Flow
DOODLE	Data-Oriented Optimization Language
DQ	Data Quality
DQM	Data Quality Management
EPC	Event-driven Process Chain
HL	High-Level
IDEF	Integration DEFinition Diagram
IT	Information Technology
LTL	Linear Temporal Logic
PAIS	Process-Aware Information System
PDI	Process Data Input
PDO	Process Data Output
Penelope	Process ENTailment from the ELicitation of Obligations and PERmissions
PSL	Process Specification Language
SBVR	Semantics of Business Vocabulary and business Rules
UML	Unified Modelling Language
WfM	Workflow Management
WS	Web Service
XPDL	XML Process Definition Language

Bibliography

- Oopus tools. "http://www.lsi.us.es/~quivir/index.php/Main/AJVarelaOPBUS", 2012.
- M. Abril López, F. Barber Sanchs, and M. A. Salido Gregorio. *Particionamiento y Resolución Distribuida Multivariable de Problemas de Satisfacción de Restricciones*. Valencia, 2007.
- F. Bacchus and F. Kabanza. Using temporal logics to express search control knowledge for planning. *Artif. Intell.*, 116(1-2):123–191, 2000.
- I. Barba. *Constraint-based Planning and Scheduling Techniques for the Optimized Management of Business Processes*. PhD thesis, University of Seville, 2012.
- I. Barba, B. Weber, C. Del Valle, and A. Jiménez Ramírez. User Recommendations for the Optimized Execution of Business Processes. *Data & Knowledge Engineering*, 86(0): 61 – 84, 2013. ISSN 0169-023X. doi: 10.1016/j.datak.2013.01.004.
- J. Becker, D. Breuker, P. Delfmann, H.-A. Dietrich, and M. Steinhorst. A runtime analysis of graph-theoretical algorithms to detect patterns in process model collections. In M. Rosa and P. Soffer, editors, *Business Process Management Workshops*, volume 132 of *Lecture Notes in Business Information Processing*, pages 489–500. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-36284-2. doi: 10.1007/978-3-642-36285-9_50. URL http://dx.doi.org/10.1007/978-3-642-36285-9_50.
- P. Benson and M. Hildebrand. *Managing Blind: A Data Quality and Data Governance Vade Mecum*. ECCMA, Bethlehem (Pennsylvania), 2012a.
- P. Benson and M. Hildebrand. Managing blind: A data quality and data governance vade mecum. Bethlehem (Pennsylvania): ECCMA, 2012b.
- I. Bermejo. I8K: Arquitectura de Servicios para la Gestión de la Calidad de los Datos: Una implementación de ISO 8000:2009-100. Master's thesis, University of Castilla-La Mancha, Ciudad Real, Spain, 2013.
- I. Bermejo and L. Parody. I8K Tool, 2013. URL <http://alarcosj.esi.uclm.es/i8k/>.
- I. Bermejo, L. Parody, I. Caballero, M. T. Gómez López, and R. M. Gasca. Gestión de calidad de datos en la combinación de actividades dentro del marco de los procesos de

- negocio. In *XVIII Jornadas de Ingeniera del Software y Bases de Datos (JISBD 2013)*, pages 195–208, 2013. ISBN 978-84-695-8310-4.
- I. Bermejo, I. Caballero, L. Parody, M. T. Gómez-López, M. Piattini, and R. Martínez Gasca. Managing data quality in master data exchange by means of iso 8000-1x0. 2014.
- I. Bider. Choosing approach to business process modeling. practical perspective. *Journal of Conceptual Modeling, Issue*, 2005.
- I. Bider, M. Khomyakov, and E. Pushchinsky. Logic of change: Semantics of object systems with active relations. *Autom. Softw. Eng.*, pages 9–37, 2000.
- D. Borrego. *Diagnostic Reasoning with Structural Analysis and Constraint Programming for Quality Improvement of Business Process Management Systems*. PhD thesis, University of Seville, 2012.
- V. Bosilj-Vuksic and V. Hlupic. Petri Nets and IDEF diagrams: Applicability and efficacy for business process modelling. *An International Journal of Computing and Informatics*, 25(1):123–133, 2001.
- J. Büchi. *The Collected Works of J. Richard Büchi*. Springer New York, 1990. ISBN 978-1-4613-8930-9. doi: 10.1007/978-1-4613-8928-6_23.
- I. Caballero, I. Bermejo, L. Parody, M. T. Gómez López, R. Martínez Gasca, and M. Piattini. I8k: An implementation of iso 8000-1x0. In *The 18th International Conference on Information Quality (ICIQ 2013)*, pages 379–393, 2013. ISBN 978-84-695-8310-4.
- C. Cappiello and B. Pernici. A methodology for information quality management in self-healing web services. In *The 11th International Conference on Information Quality (ICIQ 2006)*, pages 18–29, 2006.
- C. Cappiello, A. Caro, A. Rodríguez, and I. Caballero. An approach to design business processes addressing data quality issues. In *ECIS*, page 216, 2013.
- A. Caro, A. Rodríguez, C. Cappiello, and I. Caballero. Designing business processes able to satisfy data quality requirements. In *The 17th International Conference on Information Quality (ICIQ 2012)*, pages 31–45, 2012.
- N. Castela, J. M. Tribolet, A. Silva, and A. Guerra. Business process modeling with uml. In *ICEIS (2)*, pages 679–685, 2001.
- P. Cheeseman, B. Kanefsky, and W. M. Taylor. Where the really hard problems are. In J. Mylopoulos and R. Reiter, editors, *IJCAI*, pages 331–340. Morgan Kaufmann, 1991. ISBN 1-55860-160-0.
- T. Choco Solver. Choco 3.1.1. 2014.

- J. Chomicki. Depth-bounded bottom-up evaluation of logic program. *J. Log. Program.*, 25(1):1–31, 1995.
- B. Community. Bonita Open Solution. <http://www.bonitasoft.org/>, 2012a.
- I. Community. Intalio. <http://www.intalio.com/>, 2012b.
- P. B. Crosby. *Quality is free*. McGraw-Hill, 1979.
- I. F. Cruz. Doodle: a visual language for object-oriented databases. *SIGMOD Rec.*, 21(2):71–80, jun 1992. ISSN 0163-5808. doi: 10.1145/141484.130299. URL <http://0-doi.acm.org.fama.us.es/10.1145/141484.130299>.
- M. De Backer and M. Snoeck. Deterministic Petri net languages as business process specification language. Open access publications from katholieke universiteit leuven, Katholieke Universiteit Leuven, 2005.
- R. Dechter. *Constraint processing*. Elsevier Morgan Kaufmann, 2003. ISBN 978-1-55860-890-0.
- M. Dhring, H. A. Reijers, and S. Smirnov. Configuration vs. adaptation for business process variant maintenance: An empirical study. *Information Systems*, 39(0):108 – 133, 2014. ISSN 0306-4379. doi: <http://dx.doi.org/10.1016/j.is.2013.06.002>. URL <http://www.sciencedirect.com/science/article/pii/S0306437913000811>.
- M. Dumas, W. M. P. van der Aalst, Van der Aalst, and A. ter Hofstede, editors. *Process-Aware Information Systems: Bridging People and Software through Process Technology*. Wiley, 2005. ISBN 978-0-471-66306-5.
- L. English. *Total Quality data Management (TQdM)*. Kluwer Academic Publishers, 2001.
- K. Eshghi. Abductive planning with event calculus. In *ICLP/SLP*, pages 562–579, 1988.
- R. Ezzahir, C. Bessiere, I. Benelallam, H. Bouyakhf, and M. Belaïssaoui. Dynamic backtracking for distributed constraint optimization. In *ECAI*, pages 901–902, 2008.
- J. Fabra, V. D. Castro, P. lvarez, and E. Marcos. Automatic execution of business process models: Exploiting the benefits of model-driven engineering approaches. *Journal of Systems and Software*, 85(3):607 – 625, 2012. ISSN 0164-1212.
- D. Fahland, D. Lubke, J. Mendling, H. Reijers, B. Weber, M. Weidlich, and S. Zugal. Declarative versus imperative process modeling languages: The issue of understandability. In *Enterprise, Business-Process and Information Systems Modeling*, volume 29 of *Lecture Notes in Business Information Processing*, pages 353–366. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-01861-9.
- W. Fan and S. Weinstein. Specifying and reasoning about workflows with path constraints. In L. C. K. Hui and D. L. Lee, editors, *ICSC*, volume 1749 of *Lecture Notes in Computer Science*, pages 226–235. Springer, 1999. ISBN 3-540-66903-5.

- J. F. Fernández Narváez. Aplicación Web para la Especificación Declarativa de Procesos de Negocio. Master's thesis, University of Seville, Seville, Spain, 2013.
- E. Foundations. Epsilon. <https://www.eclipse.org/epsilon/>, 2014.
- S. Goedertier and J. Vanthienen. Designing compliant business processes with obligations and permissions. In *Business Process Management Workshops*, pages 5–14, 2006.
- S. Goedertier and J. Vanthienen. Em-bra2ce v0.1: A vocabulary and execution model for declarative business process modeling. In *Department of Decision Sciences and Information Management - KBI*, 2007.
- M. T. Gómez-López and R. Martínez Gasca. Run-time auditing for business processes data using constraints. In *Business Process Management Workshops*, pages 146–157, 2010a.
- M. T. Gómez-López and R. Martínez Gasca. Fault diagnosis in databases for business processes. In *21st International Workshop on Principles of Diagnosis, 2010*, 2010b.
- M. T. Gómez-López, R. Martínez Gasca, L. Parody, and D. Borrego. Constraint-driven approach to support input data decision-making in business process management systems. In *International Conference on Information System Development, ISD 2011*, pages 15–25. Springer, 2011.
- G. GröNer, M. Bošković, F. Silva Parreiras, and D. Gašević. Modeling and validation of business process families. *Inf. Syst.*, 38(5):709–726, July 2013. ISSN 0306-4379. doi: 10.1016/j.is.2012.11.010. URL <http://dx.doi.org/10.1016/j.is.2012.11.010>.
- T. Gschwind, J. Koehler, and J. Wong. Applying patterns during business process modeling. In M. Dumas, M. Reichert, and M.-C. Shan, editors, *Business Process Management*, volume 5240 of *Lecture Notes in Computer Science*, pages 4–19. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-85757-0. doi: 10.1007/978-3-540-85758-7_4. URL http://dx.doi.org/10.1007/978-3-540-85758-7_4.
- M. Heravizadeh, J. Mendling, and M. Rosemann. Dimensions of business processes quality (qobp). In *Business Process Management Workshops*, pages 80–91. Springer, 2009.
- K. Hirayama and M. Yokoo. Distributed partial constraint satisfaction problem. 1330: 222–236, 1997.
- S.-M. Huang, Y.-T. Chu, S.-H. Li, and D. C. Yen. Enhancing conflict detecting mechanism for web services composition: A business process flow model transformation approach. *Inf. Softw. Technol.*, 50(11):1069–1087, 2008. ISSN 0950-5849.
- ISO. ISO/DIS 8000-100: Master Data: Exchange of characteristic data: Overview. ISO, 2011a.
- ISO. ISO/DIS 8000-130: Master Data: Exchange of characteristic data: Accuracy. ISO, 2011b.

- ISO. ISO/DIS 8000-140: Master Data: Exchange of characteristic data: Completeness. ISO, 2011c.
- ISO-25012. Iso/iec 25012: Software engineering-software product quality requirements and evaluation (square)-data quality model. 2008.
- S. Jablonski and C. Bussler. *Workflow management - modeling concepts, architecture and implementation*. International Thomson, 1996. ISBN 978-1-85032-222-1.
- A. Jiménez Ramírez, I. Barba, C. Del Valle, and B. Weber. Generating multi-objective optimized business process enactment plans. In *25th International Conference on Advanced Information Systems Engineering, CAISE '13*, 2013.
- D. Knuplesch, L. T. Ly, S. Rinderle-Ma, H. Pfeifer, and P. Dadam. On enabling data-aware compliance checking of business process models. In *ER*, pages 332–346, 2010.
- O. Kopp, T. Binz, U. Breitenbücher, and F. Leymann. Bpmn4tosca: A domain-specific language to model management plans for composite applications. In *Business Process Model and Notation*, volume 125, pages 38–52, 2012. ISBN 978-3-642-33154-1.
- R. A. Kowalski and M. J. Sergot. A logic-based calculus of events. *New Generation Comput.*, 4(1):67–95, 1986.
- V. Kumar. Algorithms for constraint satisfaction problems: A survey. *AI Magazine*, 13(1):32–44, 1992.
- B. List and B. Korherr. A uml 2 profile for business process modelling. In *ER (Workshops)*, pages 85–96, 2005.
- B. List and B. Korherr. An evaluation of conceptual business process modelling languages. In *Proceedings of the 2006 ACM Symposium on Applied Computing, SAC '06*, pages 1532–1539, New York, NY, USA, 2006. ACM. ISBN 1-59593-108-2. doi: 10.1145/1141277.1141633. URL <http://doi.acm.org/10.1145/1141277.1141633>.
- D. Loshin. *Enterprises Knowledge Management: The Data Quality Approach*. Morgan Kauffman, San Francisco, CA, USA, 2001.
- L. T. Ly, S. Rinderle-Ma, and P. Dadam. Integration and verification of semantic constraints in adaptive process management systems. *Data Knowl. Eng.*, 64(1):3–23, 2008.
- L. T. Ly, D. Knuplesch, S. Rinderle-Ma, K. Goeser, H. Pfeifer, M. Reichert, and P. Dadam. Seaflows toolset - compliance verification made easy for process-aware information systems. In *Proc. CAiSE'10 Forum - Information Systems Evolution*, number 72 in LNBIP, pages 76–91. Springer, 2010. URL <http://dbis.eprints.uni-ulm.de/687/>.
- L. T. Ly, S. Rinderle-Ma, D. Knuplesch, and P. Dadam. Monitoring business process compliance using compliance rule graphs. In *19th International Conference on Cooperative Information Systems (CoopIS 2011)*, number 7044 in LNCS, pages 82–99. Springer, 2011.

- F. M. Maggi, M. Montali, M. Westergaard, and W. van der Aalst. Monitoring Business Constraints with Linear Temporal Logic: An Approach Based on Colored Automata. In *Proc. of BPM*, LNCS. Springer-Verlag, 2011a.
- F. M. Maggi, M. Westergaard, M. Montali, and W. van der Aalst. Runtime Verification of LTL-Based Declarative Process Models. In *Proc. of RV*, LNCS. Springer-Verlag, 2011b.
- F. M. Maggi, M. Westergaard, W. M. P. van der Aalst, F. Staff, M. Pesic, and H. Schonberg. Declare tool. 2014.
- R. Manual. Jsolver 2.1. 2003.
- D. B. María Teresa Gómez-López and R. M. Gasca. Data state description for the migration to activity-centric business process model maintaining legacy databases. In *17th International Conference on Business Information Systems (BIS 2014)*, 2014.
- A. Marotta, L. González, and R. Ruggia. A quality aware service-oriented web warehouse platform. In *Proceedings of the 2012 Joint EDBT/ICDT Workshops*, EDBT-ICDT '12, pages 29–32, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1143-4. doi: 10.1145/2320765.2320783.
- K. Marriott and P. Stuckey. *Programming with Constraints: An Introduction*. Adaptive Computation and Machine. MIT Press, 1998. ISBN 9780262133418. URL <http://books.google.es/books?id=jBYAleHTldsC>.
- J. Mendling and M. Weidlich, editors. *Business Process Model and Notation - 4th International Workshop, BPMN 2012, Vienna, Austria, September 12-13, 2012. Proceedings*, volume 125 of *Lecture Notes in Business Information Processing*, 2012. Springer. ISBN 978-3-642-33154-1.
- A. Meyer, S. Smirnov, and M. Weske. Data in business processes. *EMISA Forum*, 31(3): 5–31, 2011.
- A. Meyer, L. Pufahl, D. Fahland, and M. Weske. Modeling and enacting complex data dependencies in business processes. In *BPM*, pages 171–186, 2013.
- M. Montali, F. Chesani, P. Mello, and F. M. Maggi. Towards data-aware constraints in declare. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing, SAC '13*, pages 1391–1396. ACM, 2013.
- K. Namiri and N. Stojanovic. Using control patterns in business processes compliance. In M. Weske, M.-S. Hacid, and C. Godart, editors, *Web Information Systems Engineering - WISE 2007 Workshops*, volume 4832 of *Lecture Notes in Computer Science*, pages 178–190. Springer Berlin - Heidelberg, 2007. ISBN 978-3-540-77009-1.
- NIST. *Process Specification Language (PSL) Version 2.1*. National Institute of Standards and Technology (NIST), 2004.

- B. V. Nuffelen and A. C. Kakas. A-system: Declarative programming with abduction. In T. Eiter, W. Faber, and M. Truszczynski, editors, *LPNMR*, volume 2173 of *Lecture Notes in Computer Science*, pages 393–396. Springer, 2001. ISBN 3-540-42593-4.
- OASIS. *Business Process Execution Language for Web Services*. OASIS Standard, 2005.
- OASIS. *Web Services Business Process Execution Language (BPEL) Version 2.0*. OASIS Standard, 2007.
- I. Ognjanovic, B. Mohabbati, D. Gasevic, E. Bagheri, and M. Boskovic. A metaheuristic approach for the configuration of business process families. In L. E. Moser, M. Parashar, and P. C. K. Hung, editors, *IEEE SCC*, pages 25–32. IEEE, 2012. ISBN 978-1-4673-3049-7.
- OMG. *OMG Model Driven Architecture*. OMG Standard, 2003.
- OMG. Unified Modeling Language: Superstructure version 2.0. <http://www.uml.org/>, 2005.
- OMG. Semantics of business vocabulary and business rules (sbvr). 2008.
- OMG. *BPMN 2.0 by Example. Version 1.0 (non-normative)*. OMG Standard, 2011a.
- OMG. *Business Process Model and Notation (BPMN) Version 2.0*. Object Management Group Standard, 2011b.
- M. P. Papazoglou and W.-J. van den Heuvel. Service oriented architectures: approaches, technologies and research issues. *VLDB J.*, 16(3):389–415, 2007.
- L. Parody. CombiS-BP Editor, 2013. URL <http://www.lsi.us.es/~quivir/index.php/Main/LParodyCombi-BP>.
- L. Parody, M. T. Gómez-López, R. Martínez Gasca, and D. Borrego. Resolución de acuerdos en procesos de negocio para multiprocesos software usando programación con restricciones distribuidas. In *Workshop de Apoyo a la Decisión en Ingeniería del Software (ADIS10)*, volume 4, pages 23–34, 2010.
- L. Parody, M. T. Gómez-López, R. Martínez Gasca, and D. Borrego. Using distributed csp to model business processes agreement in software multiprocess. In *3rd International Conference on Agents and Artificial Intelligence (ICAART'11)*, volume 2, pages 434–438, 2011a. ISBN 978-989-8425-41-6.
- L. Parody, M. T. Gómez-López, R. Martínez Gasca, and A. J. Varela-Vaca. An approach for optimization agreements in business processes based on web services. In *17th International Business Information Management Association Conference (IBIMA Conference)*, pages 183–194, 2011b. ISBN 978-0-9821489-6-6.
- L. Parody, M. T. Gómez-López, and R. Martínez Gasca. Extending bpmn 2.0 for modelling the combination of activities that involve data constraints. In Mendling and Weidlich (2012), pages 68–82. ISBN 978-3-642-33154-1.

- L. Parody, M. T. Gómez-López, R. Martínez Gasca, and A. J. Varela-Vaca. Improvement of Optimization Agreements in Business Processes involving Web Services. *Communications of the IBIMA*, 2012, 2012b.
- L. Parody, M. T. Gómez-López, and R. Martínez Gasca. Data-oriented declarative language for optimizing business processes. In *22nd International Conference on Information Systems Development (ISD2013)*, page To appear, 2013a.
- L. Parody, M. T. Gómez-López, and R. Martínez Gasca. Decision-making sub-process to obtain the optimal combination of input data in business processes. In *IX Jornadas de Ciencia e Ingeniería de Servicios (JCIS 2013)*, pages 17–31, 2013b. ISBN 978-84-695-8351-7.
- L. Parody, M. T. Gómez-López, R. Martínez Gasca, and A. J. Varela-Vaca. Combis-bp editor: Combining declarative and imperative languages in bp modelling. In *Seventh IEEE International Conference on Research Challenges in Information Science*, pages 663–664, 2013c. ISBN 978-1-4673-2914-9.
- L. Parody, M. T. Gómez-López, I. Bermejo, I. Caballero, R. Martínez Gasca, and M. Piattini. Pais-dq: Extending process-aware information systems to support data quality in pais life-cycle. 2014a.
- L. Parody, M. T. Gómez-López, and R. Martínez Gasca. Configuration of an imperative business process according to data dependency aspects. 2014b.
- L. Parody, M. T. Gómez-López, and R. Martínez Gasca. Optimization of outcome data of business processes. from declarative to imperative process models. 2014c.
- D. Peixoto, V. Batista, A. Atayde, E. Borges, R. Resende, and C. Pádua. A comparison of bpmn and uml 2.0 activity diagrams. In *VII Simposio Brasileiro de Qualidade de Software*, volume 56, 2008.
- M. Pesic and W. M. P. van der Aalst. A declarative approach for flexible business processes management. In J. Eder and S. Dustdar, editors, *Business Process Management Workshops*, volume 4103 of *Lecture Notes in Computer Science*, pages 169–180. Springer, 2006. ISBN 3-540-38444-8.
- M. Pesic, H. Schonenberg, and W. M. P. van der Aalst. Declare demo: A constraint-based workflow management system. In *BPM (Demos)*, 2009.
- C. A. Petri. Fundamentals of a Theory of Asynchronous Information Flow. In *IFIP Congress*, pages 386–390, 1962.
- C. J. Petrie. *Automated Configuration Problem Solving*. Springer Publishing Company, Incorporated, 2012. ISBN 1461445310, 9781461445319.
- L. Pipino, Y. Lee, and R. Wang. Data quality assessment. *Communications of the ACM*, 45(4):211–218, Apr. 2002. ISSN 0001-0782. doi: 10.1145/505248.506010. URL <http://doi.acm.org/10.1145/505248.506010>.

- Princeton University. WordNet: A lexical database for English, 2014. URL <http://wordnet.princeton.edu/>. Accessed on the 24th of February of 2014.
- T. Redman. the impact of poor data quality on the typical enterprise. *Commun. ACM*, 41(2):79–82, 1998.
- A. Rodríguez, E. Fernández-Medina, and M. Piattini. Towards a uml 2.0 extension for the modeling of security requirements in business processes. In S. Fischer-Hbner, S. Furnell, and C. Lambrinoudakis, editors, *Trust and Privacy in Digital Business*, volume 4083 of *Lecture Notes in Computer Science*, pages 51–61. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-37750-4. doi: 10.1007/11824633_6. URL http://dx.doi.org/10.1007/11824633_6.
- A. Rodríguez, A. Caro, C. Cappiello, and I. Caballero. A bpmn extension for including data quality requirements in business process modeling. In Mendling and Weidlich (2012), pages 116–125. ISBN 978-3-642-33154-1.
- W. D. Roover, F. Caron, and J. Vanthienen. A prototype tool for the event-driven enforcement of sbvr business rules. In F. Daniel, K. Barkaoui, and S. Dustdar, editors, *Business Process Management Workshops (1)*, volume 99 of *Lecture Notes in Business Information Processing*, pages 446–457. Springer, 2011.
- M. L. Rosa, M. Dumas, A. H. ter Hofstede, and J. Mendling. Configurable multi-perspective business process models. *Information Systems*, 36(2):313 – 340, 2011. ISSN 0306-4379. doi: <http://dx.doi.org/10.1016/j.is.2010.07.001>. URL <http://www.sciencedirect.com/science/article/pii/S0306437910000633>. Special Issue: Semantic Integration of Data, Multimedia, and Services.
- F. Rossi, P. v. Beek, and T. Walsh. *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. Elsevier Science Inc., New York, NY, USA, 2006. ISBN 0444527265.
- W. Runte. Modelling and solving configuration problems on business processes using a multi-level constraint satisfaction approach. In W. Abramowicz, L. A. Maciaszek, R. Kowalczyk, and A. Speck, editors, *BPSC*, volume 147 of *LNI*, page 237. GI, 2009. ISBN 978-3-88579-241-3.
- W. Runte and M. E. Kharbili. Constraint checking for business process management. In S. Fischer, E. Maehle, and R. Reischuk, editors, *GI Jahrestagung*, volume 154 of *LNI*, pages 4093–4103. GI, 2009. ISBN 978-3-88579-248-2.
- N. Russell, W. M. P. van der Aalst, A. H. M. ter Hofstede, and P. Wohed. On the suitability of uml 2.0 activity diagrams for business process modelling. In *Proceedings of the 3rd Asia-Pacific Conference on Conceptual Modelling - Volume 53*, APCCM '06, pages 95–104, Darlinghurst, Australia, Australia, 2006. Australian Computer Society, Inc. ISBN 1-920-68235-X. URL <http://dl.acm.org/citation.cfm?id=1151855.1151866>.

- I. Rychkova and S. Nurcan. Towards adaptability and control for knowledge-intensive business processes: Declarative configurable process specifications. In *HICSS*, pages 1–10, 2011.
- I. Rychkova, G. Regev, and A. Wegmann. High-level design and analysis of business processes: the advantages of declarative specifications. In O. Pastor, A. Flory, and J.-L. Cavarero, editors, *RCIS*, pages 99–110. IEEE, 2008a. ISBN 978-1-4244-1677-6.
- I. Rychkova, G. Regev, and A. Wegmann. Using declarative specifications in business process design. *IJCSA*, 5(3b):45–68, 2008b.
- S. W. Sadiq, M. E. Orlowska, and W. Sadiq. Specification and validation of process constraints for flexible workflows. *Information Systems*, 30(5):349 – 378, 2005. ISSN 0306-4379. doi: {10.1016/j.is.2004.05.002}.
- S. W. Sadiq, G. Governatori, and K. Namiri. Modeling control objectives for business process compliance. In *Proceedings of the 5th international conference on Business process management*, BPM’07, pages 149–164. Springer-Verlag, 2007. ISBN 3-540-75182-3, 978-3-540-75182-3.
- K. Salimifard and M. Wright. Petri net-based modelling of workflow systems: An overview. *European Journal of Operational Research*, 134(3):664 – 676, 2001. ISSN 0377-2217. doi: [http://dx.doi.org/10.1016/S0377-2217\(00\)00292-7](http://dx.doi.org/10.1016/S0377-2217(00)00292-7). URL <http://www.sciencedirect.com/science/article/pii/S0377221700002927>.
- N. G. Saoussen Cheikhrouhou, Slim Kallel and M. Jmaiel. The temporal perspective in business process modeling : An evaluative survey and research challenges. Open access publications, Unit de Recherche en developpement et contrle d’applications distribues (REDCAD), Laboratoire d’analyse et d’architecture des systmes [Toulouse] (LAAS) and Institut national Des Sciences Appliques de Toulouse (INSA Toulouse), 2013.
- P. Sawyer, R. Mazo, D. Diaz, C. Salinesi, and D. Hughes. Using constraint programming to manage configurations in self-adaptive systems. *Computer*, 45(10):56–63, 2012. ISSN 0018-9162. doi: <http://doi.ieeecomputersociety.org/10.1109/MC.2012.286>.
- M. Shanahan. Event calculus planning revisited. In S. Steel and R. Alami, editors, *ECP*, volume 1348 of *Lecture Notes in Computer Science*, pages 390–402. Springer, 1997. ISBN 3-540-63912-8.
- M.-C. Silaghi and M. Yokoo. Adopt-ing: unifying asynchronous distributed optimization with asynchronous backtracking. *Autonomous Agents and Multi-Agent Systems*, 19(2): 89–123, 2009.
- S. Smirnov, M. Weidlich, J. Mendling, and M. Weske. *Action patterns in business process models*. Springer, 2009.
- S. X. Sun, J. L. Zhao, J. F. Nunamaker, and O. R. L. Sheng. Formulating the data-flow perspective for business process management. *Info. Sys. Research*, 17(4):374–391, dec

2006. ISSN 1526-5536. doi: 10.1287/isre.1060.0105. URL <http://dx.doi.org/10.1287/isre.1060.0105>.
- A. Team. Activity BPM Platform. <http://www.activiti.org/>, 2012.
- R. K. Thiagarajan and M. Stumptner. Service composition with consistency-based match-making: A csp-based approach. In *ECOWS*, pages 23–32. IEEE Computer Society, 2007.
- V. Torres and V. Pelechano. Building business process driven web applications. In S. Dustdar, J. L. Fiadeiro, and A. P. Sheth, editors, *Business Process Management*, volume 4102 of *Lecture Notes in Computer Science*, pages 322–337. Springer, 2006. ISBN 3-540-38901-6.
- A. Tsai, J. Wang, W. Tepfenhart, and D. Rosea. Epc workflow model to wifa model conversion. In *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, volume 4766/2007, pages 2758 – 2763. IEEE International Conference on Systems, Man and Cybernetics, 2006. SMC '06., 2006. ISBN 1-4244-0099-6.
- W. M. P. Van der Aalst. Formalization and verification of event-driven process chains. *Information & Software Technology*, 41(10):639–650, 1999.
- W. M. P. van der Aalst. Business process management demystified: A tutorial on models, systems and standards for workflow management. *Lectures on Concurrency and Petri Nets*, pages 21–58, 2004.
- W. M. P. van der Aalst. A decade of business process management conferences: Personal reflections on a developing discipline. In *Business Process Management*, volume 7481 of *Lecture Notes in Computer Science*, pages 1–16. Springer Berlin / Heidelberg, 2012. ISBN 978-3-642-32884-8.
- W. M. P. van der Aalst and C. Stahl. *A Petri Net-Oriented Approach*. The MIT Press, 2011. ISBN 978-0-262-01538-7.
- W. M. P. van der Aalst, A. H. M. ter Hofstede, and M. Weske. Business process management: A survey. In W. M. P. van der Aalst, A. H. M. ter Hofstede, and M. Weske, editors, *Business Process Management*, volume 2678 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2003. ISBN 3-540-40318-3.
- W. M. P. van der Aalst, M. Weske, and D. Grnbauer. Case handling: A new paradigm for business process support. *Data and Knowledge Engineering*, 53, 2005.
- A. J. Varela-Vaca, R. Martínez Gasca, and S. Pozo. Opbus: Risk-aware framework for the conformance of security-quality requirements in business processes. In *SECRYPT*, pages 370–374, 2011.
- C. Venera Geambasu. BPMN vs. UML Activity Diagram for Business Process Modeling. *Journal of Accounting and Management Information Systems*, 11(4):637–651, December 2012. URL <http://ideas.repec.org/a/ami/journal/v11y2012i4p637-651.html>.

- R. Y. Wang. A product perspective on total data quality management. *Communications of the ACM*, 2(41):58–65, 1998.
- B. Weber, S. W. Sadiq, and M. Reichert. Beyond rigidity - dynamic process lifecycle support. *Computer Science - R&D*, 23(2):47–65, 2009.
- M. Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer, 2007. ISBN 978-3-540-73521-2.
- WfMC. *XML Process Definition Language) Version 2.1*. The Workflow Management Coalition (WfMC), 2004.
- Y. Wu and P. Doshi. Making bpel flexible - adapting in the context of coordination constraints using ws-bpel. pages 423–430, 2008. doi: <http://dx.doi.org/10.1109/SCC.2008.71>.
- M. Yokoo and K. Hirayama. Algorithms for distributed constraint satisfaction: A review. In *AAAI*, volume 3, pages 185–207, 2000.
- M. Yokoo, E. H. Durfee, T. Ishida, and K. Kuwabara. The distributed constraint satisfaction problem: Formalization and algorithms. *IEEE Transactions on knowledge and data engineering*, 10(5):673–685, 1998.