PhD Dissertation

International Doctorate School in Information and Communication Technologies

DISI - University of Trento

Semantic Annotation of Business Process Models

Chiara Di Francescomarino

Advisor:

Dr. Paolo Tonella

Fondazione Bruno Kessler (FBK)

April 13, 2011

To my Family
and my Angels

Abstract

In the last decades, business process models have increasingly been used by companies with different purposes, such as documenting enacted processes or enabling and improving the communication among stakeholders (e.g., designers and implementers). Aside from the differences, all the roles played by process models involve human actors (e.g., business designers, business analysts, re-engineers) and hence demand for readability and ease of use, beyond correctness and reasonable completeness. It often happens, however, that process models are large and intricate, thus resulting potentially difficult to understand and to manage.

In this thesis we propose some techniques aimed at supporting business designers and analysts in the management of business process models. The core of the proposal is the enrichment of process models with semantic annotations from domain ontologies and the formalization of both structural and domain information in a shared knowledge base, thus opening to the possibility of exploiting reasoning for supporting business experts in their work. In detail, this thesis investigates some of the services that can be provided on top of the process semantic annotation, as for example, the automatic verification of process constraints, the automated querying of process models or the semi-automatic mining, documentation and modularization of crosscutting concerns. Moreover, special care is devoted to support designers and analysts when process models are not available or they have to be semantically annotated. Specifically, an approach for recovering process models from (Web) applications and some metrics for evaluating the understandability of the recovered models are investigated. Techniques for suggesting candidate semantic annotations are also proposed. The results obtained by applying the presented techniques have been validated by means of case studies, performance evaluations and empirical investigations.

Keywords
Business Processes, Semantic Annotation, Reverse Engineering, Crosscutting Concerns, Constraint Verification

Acknowledgements

I would like to thank all the people who supported me, in different ways, during the PhD.

First of all, my full gratitude to my advisor Paolo Tonella, who guided me with patience, encouraged me and provided me with precious suggestions and advices.

I would like to thank Alessandro Marchetto, Chiara Ghidini, Marco Rospocher and Luciano Serafini for the interesting and fruitful collaboration we carried out. They stimulated the investigation of new research fields and allowed me to look at things from different perspectives. A special thank to Marco for the amount of time he spent in answering my questions and clarifying my doubts. A warm thank to Alessandro for the interesting work experiences we shared, for his advices and for his invaluable support.

Thanks to Professor Michele Flammini and Maria Luisa Villani who encouraged me in starting this new experience and to Professor Massimiliano Di Penta who introduced me in the world of empirical studies.

Thanks to Anna Perini and Angelo Susi: with their presence and advices, they supported me in facing difficulties and in the everyday life.

Thanks to all my friends and colleagues who shared with me the whole or part of this road. In alphabetical order by name: Alberto Siena, Andrea Avancini, Andrea Mattioli, Birhanu Eshete, Carlos Cares, Cu Du Nguyen, Cuong To Tu, Elena Cardillo, Fitsum Kifetew, Francis Palma, Karen Najera Hernandez, Komminist Sisai Weldemariam, Ismênia Galvão, Leonardo Leiria Fernandes, Luca Sabatucci, Mariano Ceccato, Martin Homola, Mathew Joseph, Matteo Ceriotti, Mirko Morandini, Nauman Qureshi, Roberto Tiella, Sapna Poranaseri, Sepideh Ghanavati, Shiva Vafadar and Surafel Lemma. Their presence, their ideas, as well as discussions and experiences we shared created a welcoming, stimulating and pleasant environment of work and life.

Moreover, my gratitude to all the persons who spent their time participating in the empirical studies.

Grazie ai miei Angeli che, con la loro costante presenza, sono stati e sono sempre accanto a me in tutti i momenti importanti, sia in quelli di gioia che in quelli di difficoltà.

Infine, un grazie speciale alla mia mamma, al mio papà e a mia sorella per il loro aiuto ed il loro supporto, per i loro continui incoraggiamenti, per la forza che ogni giorno mi danno e per il loro infinito amore.

Contents

Abstract **e**

Acknowledgements **g**

1 Introduction **1**

1.1 Context . . . 1

1.2 Problem . . . 1

1.3 Solution . . . 2

1.4 Innovative Aspects . . . 6

1.5 Publications . . . 7

1.6 Structure of the Thesis . . . 8

2 Background **11**

2.1 Business Process Management . . . 11

2.1.1 Business Process Models . . . 11

2.2 Semantic Business Process Management . . . 15

2.2.1 Semantic Annotation of Process Models . . . 15

2.2.2 Ontologies . . . 16

3 Reverse Engineering of Business Process Models **19**

3.1 Reverse Engineering of Business Processes . . . 20

3.1.1 Background . . . 21

3.1.2 GUI-based Reverse Engineering . . . 24

3.1.3 Dynamic Process Extraction . . . 26

3.1.4 Process Clustering . . . 30

3.1.5 Cluster Labelling . . . 39

3.1.6 The tool 41
3.1.7 Reverse Engineering Technique Evaluation 43
3.2 Understandability Metrics 52
3.2.1 Process Metrics 53
3.2.2 Experimental Study 63

4 Business Process Semantic Annotation 85
4.1 Semantic Annotation of BPMN Process Models 86
4.1.1 Enriching BPMN Processes with Semantic Annotations 86
4.1.2 Formalizing Semantically Annotated BPMN Processes 87
4.1.3 Automatically encoding a BPD into an Abox 92
4.2 Semantic Annotation Suggestions 94
4.2.1 Background 95
4.2.2 Domain Ontology Analysis 97
4.2.3 Business Process Semantic Annotation Suggestions 105
4.2.4 Domain Ontology Extension 109
4.2.5 Automatic Suggestion Evaluation 115

5 Constraint Verification 123
5.1 Process Requirement Specification 124
5.1.1 An Explanatory Example 124
5.1.2 Merging Axioms 127
5.1.3 Process Specific Constraints 130
5.1.4 User-friendly Constraint Representation 148
5.2 Constraint Verification 149
5.2.1 Compatibility Checking of Process Constraints 150
5.2.2 Constraints Verification over an Annotated BPD 152
5.3 Constraint Checking Performance Evaluation 153

6 Crosscutting Concern Documentation 157
6.1 Concern Querying 158
6.1.1 BPMN VQL 159
6.1.2 BPMN VQL Evaluation 166
6.2 Crosscutting Concern Mining 173
6.2.1 Crosscutting Concern Mining 174

6.2.2 Crosscutting Concern Mining Evaluation 176
6.3 Concern Documentation 186

7 Business Process Aspectization 189
7.1 Semantically enhanced aspects 190
7.1.1 Aspect Oriented Programming 191
7.1.2 BPMN VRL 192
7.2 Exception Handling Aspectization 209
7.2.1 Using semantic constraints to support aspect definition 212
7.3 Performance Evaluation 214

8 Experimental Results:
BPMN VQL Empirical Evaluation 217
8.1 Experiment Definition, Planning and Design 218
8.1.1 Goal of the Study and Research Questions 218
8.1.2 Context 220
8.1.3 Design, Material and Procedure 221
8.1.4 Variables 223
8.2 Experimental Results 227
8.2.1 Data Analysis 228
8.2.2 Cofactors 233
8.3 Threats to Validity 240
8.3.1 Conclusion Validity 240
8.3.2 Internal Validity 240
8.3.3 Construct Validity 241
8.3.4 External Validity 241
8.4 Discussion 242

9 Related Works 247
9.1 Reverse Engineering and Understandability Metrics 247
9.1.1 Reverse Engineering 247
9.1.2 Process Understandability 250
9.2 Business Process Semantic Annotation 251
9.3 Constraint Verification on Business Processes 254
9.4 Crosscutting Concerns 258
9.4.1 Crosscutting Concerns in Software 258

9.4.2 Crosscutting concerns and processes 261
9.5 Business Process Exception Handling 264
9.6 Visual Process Query Language Evaluation 265

10 Conclusions and Future Works 267

Bibliography 271

A Empirical Study Pre-questionnaire 291

B Empirical Study Post-questionnaire 292

List of Tables

3.1 Initially recovered process data 46
3.2 Clustered sub-process and label quality 48
3.3 Experiment design 65
3.4 Descriptive statistics of the metric measures 65
3.5 Descriptive statistics of the normalized metric measures 66
3.6 Descriptive statistics about MOs 69
3.7 Correlation between process metrics and MOs 73
3.8 Correlation between pairs of metrics 74
3.9 Analysis of the modellers' opinions 76
3.10 Regression models 79
3.11 Accuracy of the predictions 80
3.12 Accuracy of understandable/non-understandable predictions 81
3.13 Understandability and process metrics: flat versus modularized models . . 82

4.1 Mapping from $MGCS'$ to $WNPOS$: function $wnPOS$. 97
4.2 Example values for the RDA 100
4.3 An example of possible problems raised by RDA 101
4.4 Example values for the CDA 103
4.5 Application of DBDA to search 104
4.6 ics_{Lin} between an activity label and an ontology concept 108
4.7 Five ontology concepts most similar to the label "Choose a product group" 108
4.8 Heuristic rules to create a candidate ontology 110
4.9 Ranking of BDO concepts similar to the concept group 113
4.10 Three composed concepts most similar to the label "Choose a product group" 114
4.11 Disambiguation algorithm application to ontology in Figure 4.11 116
4.12 Disambiguation algorithm application to the ontology in Figure 4.12 (left) 117
4.13 Disambiguation algorithm application to the ontology in Figure 4.12 (right) 117

4.14 Case study data 119

5.1 Merging axiom patterns 129
5.2 Requirement Patterns 148
5.3 Perofrmance Evaluation Results I 154
5.4 Performance Evaluation Results II 155

6.1 Visual Query Language Comparison 169
6.2 BPMN VQL performance 172
6.3 BPMN VQL performance on non-preprocessed ontologies 172
6.4 On-line purchase case study: top ranking of the FCA-concepts 180
6.5 Issue-voting case study: top ranking of the FCA-concepts 185

7.1 Aspectization and Validation performance 215

8.1 Study balanced design 221
8.2 Dependent variable description 225
8.3 Descriptive statistics for the *Query Understandability* assignment 230
8.4 Results *Query Understandability* assignment 230
8.5 Descriptive statistics of the *Query Execution* assignment 231
8.6 Results *Query Execution* assignment 233
8.7 Summary table of the cofactors considered in the experiment evaluation . . 234
8.8 Cofactor analysis related to **RQ1** 234
8.9 Cofactor analysis related to **RQ2** 236
8.10 Personal judgement survey results 239

List of Figures

1.1 Thesis flow 9

2.1 An example of Business Process Diagram 13

3.1 Flow Analysis Framework 23
3.2 Application of the Flow Analysis Framework for dominators 25
3.3 Business process reverse engineering approach 26
3.4 An example of user navigation trough web pages 27
3.5 Business process recovery algorithm 28
3.6 Fragment of the process example 30
3.7 Loop clustering examples 32
3.8 Alternative clustering example 33
3.9 Hill climbing algorithm 37
3.10 Dependency clustering example 37
3.11 Term-based analysis example 38
3.12 Example of TFIDF values 41
3.13 The recovered Softslate process 51
3.14 $PM_{example1}$(top) and $PM_{example2}$(bottom) process models 56
3.15 Distribution of the measured normalized metrics 69
3.16 Distribution of MOs 69
3.17 Softslate, Communicart and Erol boxplots 71
3.18 Plot of some metrics and modellers' opinions 72
3.19 Plot of predicted and actual MO values 80

4.1 An example of a semantically annotated BPMN process 88
4.2 The Business Process Knowledge Base 89
4.3 The graphical elements of BPMNO 89
4.4 A sub-process for the cart management in an on-line purchase process . . . 91

4.5 The encoding of the to_manage_cart sub-process in an OWL Abox 92
4.6 A fragment of the mapping file 93
4.7 Information extracted by MINIPAR 96
4.8 Examples of ontologies 100
4.9 Automatically suggested position for concept group 114
4.10 Candidate ontology skeleton construction 115
4.11 An example of an ontology with a structure different from the one of WordNet 116
4.12 Examples of ontologies similar to WordNet 117
4.13 Extracts of the OntoSem ontology 120

5.1 A portion of the on-line shopping business process diagram. 125
5.2 Diagrams with equivalent behaviour 127
5.3 The classification of intermediate events 142
5.4 Two views of the functionalities provided by BP-MoKi 149
5.5 Example of BP-MoKi "annotates only" merging axiom 150
5.6 Explanation generation 151

6.1 An example of a semantically annotated BPMN process. 161
6.2 Example of a query using semantic annotations 162
6.3 Example of a query using BPMN, stereotypes and semantic annotations . . 162
6.4 Example of a query using logical operators 163
6.5 Example of a query using the OR operator 164
6.6 Example of a query using the NOT operator 164
6.7 Example of a query using the PATH operator 166
6.8 Example of a query using the NEST operator 167
6.9 Example of a query using the DOR operator 168
6.10 Examples of FCA-concepts and concept lattice 175
6.11 Semantically annotated on-line purchase process 178
6.12 On-line purchase case study: concept lattice 179
6.13 Precision, recall and F-Measure of the two case studies 181
6.14 Ontology for the annotation of an on-line purchase process 182
6.15 Semantically annotated e-mail vote process 184
6.16 Visual query for the "user preference management" concern 186

7.1 Simple example of BPMN VRL rule adding behaviour 193
7.2 Simple example of BPMN VRL rule removing behaviour 193
7.3 Example process 195

7.4 A portion of the On-line shopping business process diagram. 197
7.5 Example of a BPMN VRL rule adding behaviour 198
7.6 Example of a BPMN VRL rule removing behaviour 199
7.7 Example of a BPMN VRL rule adding and removing behaviour 200
7.8 Example of a BPMN VRL rule using the BPMN VQL OR operator 202
7.9 Example of a BPMN VRL rule using the BPMN VQL NOT operator 203
7.10 Example of a BPMN VRL rule using the BPMN VQL PATH operator 205
7.11 Example of a BPMN VRL rule using the BPMN VQL NEST operator 207
7.12 Example of a BPMN VRL rule with "update" parts of different cardinality . 208
7.13 Example of two BPMN VRL rules for solving problems of cardinality 208
7.14 Product unavailability aspect 210
7.15 Compulsory log-in failure aspect 212
7.16 Partial view of the woven on-line Shop process 213

8.1 Detailed study procedur 224
8.2 Boxplots related to the *Query Understandability* assignment 229
8.3 Boxplots related to the *Query Execution* assignment 232
8.4 Boxplots related to the time spent in matching queries 235
8.5 Plots related to the perceived effort in matching querie 236
8.6 Plots related to the perceived effort in executing queries 237
8.7 Plots related to the perceived effort in specification understanding 238

Chapter 1

Introduction

In the current chapter we introduce the work presented in this thesis, by starting from the description of its context (Section 1.1) and the problems arising in this scenario (Section 1.2). Moreover, we present an overview of the solutions we propose to deal with these problems throughout the thesis (Section 1.3) and their contributions to the state of the art (Section 1.4). Finally, Section 1.5 provides the list of published papers containing the material presented in the thesis and Section 1.6 a description of the thesis structure.

1.1 Context

Starting from the process orientation trend that involved companies in the 1990s, the importance and the spread of Business Process Management (and hence of business processes) in companies' organization has notably increased. In this scenario, business process models have played a fundamental role. In fact, beyond being the outcome of the first step of the business process life-cycle (i.e., they are designed to be implemented by developers), they also act as a means for facilitating the communication among stakeholders (including business experts) and as a form of documentation of the process enacted by the company, thus providing a faster response to analysis needs. All these different uses of business process models, however, share a common goal: they mainly aim at meeting business people's (i.e., designers, analysts, business men, re-engineers) needs.

1.2 Problem

Business process models, depending on their purpose, could be very large, involve many different business perspectives and weave several distinct business concerns. As a consequence, in these cases, they can be difficult to read, understand, analyse, design or re-engineer, thus going against their initial objectives. Business process description and modelling languages provide means for describing the main views, which define the principal decomposition of the business process, such as the control flow, data flow and performers. However, they do not allow to describe concerns that are scattered across the process and tangled with other concerns (crosscutting concerns), which, hence, are often left implicit. Moreover, there is a lack of mechanisms supporting business experts to check whether the requirements they specify over the models are verified, as well as to handle exceptions without making processes complicated and intricate.

Stakeholders in charge of dealing with business process models could be supported by exploiting reasoning services and providing them with automatic tools. To this purpose, however, semantic knowledge is required.

The benefits of having formal semantic information in business processes would be twofold: on one side, it would facilitate the communication among stakeholders, by providing a clear business and domain semantics to the process elements and, on the other, it would allow information systems to acquire knowledge, thus enabling reasoning and providing stakeholders with automatic tools for their support.

Despite the advantages and services semantic information could provide to business experts, it may happen that: (i) a business process model documenting the actual realization of processes enacted in companies does not always exist; (ii) even if it exists, enriching it with semantic knowledge can be an expensive activity for business designers and analysts. Business designers and analysts, hence, should be supported also in the preparatory phase, i.e., in process model recovery (if the model is not available) and in its semantic enrichment.

1.3 Solution

The main goal of this thesis is to support people working with business process models in their management (i.e., comprehension, design, analysis), thus attempting to limit the drawbacks they face (as those presented in the previous section). In detail, to this aim, aware of the lack of semantic information in business process models, we propose to enrich the models with a formal semantics by annotating process elements with domain concepts. The added knowledge aims at supporting business designers and analysts, by clarifying the process model semantics and by enabling reasoning services, such as constraint verification, process querying, crosscutting concern retrieval and documentation, as well as aspect and exception handling management in process models.

In this thesis, we propose a framework in which the structural and the semantic information contained in the enriched process models are formalized and included in a common knowledge base, the Business Process Knowledge Base (BPKB). The specific process model information, however, represents only the changeable part (*assertional knowledge*) of the knowledge base, i.e., the process-specific instances populating the knowledge base. The BPKB classes that are instantiated in the assertional knowledge (including the classes describing the process structure, the classes related to the business domain knowledge and a set of axioms constraining their relationships) represent instead the unchangeable part of the knowledge base, i.e., the *terminological knowledge*.

In detail, in this thesis, we focus on BPMN (Business Process Modeling Notation) processes and, in particular, on BPMN processes enriched with semantic annotations belonging to one or more domain ontologies. The terminological part of the knowledge base contains classes from a BPMN ontology (formalized starting from the BPMN specifications [66]), as well as from one or more domain ontologies (specializations of an upper level ontology [65]) including the concepts used for the semantic annotation of the specific process model.

By taking advantage of the business domain knowledge added to business process models and of the BPKB formalization, we investigated some of the possible services for the business experts' support. In detail, on the top of the process semantic annotation and formalization we considered:

- the automatic verification of process constraints. It supports business experts when a number of constraints have to be enforced by the process model, especially in cases of collaborative environments, for large processes and when maintenance or restructuring operations are required. The proposed approach is theoretically grounded on the Description Logics. Once the semantically annotated process model has been encoded in the BPKB, the constraints have been formulated as Description Logics axioms and have been added to the knowledge base, constraint verification is realized by checking the consistency of the BPKB. A classification of some of the constraint categories, and of the corresponding pattern translations in Description Logics, is reported in this thesis.

- the possibility of automatically querying semantically annotated business process models by formulating queries in a visual query language, BPMN VQL (BPMN Visual Query Language). It allows business designers and analysts to retrieve particular concerns characterized by specific business and structural features. Moreover, BPMN

VQL is an easy-to-use language for querying BPMN processes with a syntax, close to the BPMN itself, that aims at reducing the learning effort of BPMN experts. Queries, formulated in BPMN VQL, are automatically executed by exploiting the process model formalization into a knowledge base and their results are visually highlighted in the process model.

- an approach for semi-automatically mining and documenting crosscutting concerns. Crosscutting concerns are those concerns that are scattered across the process and tangled with other concerns, thus resulting difficult to locate and analyse mainly in large processes. Business designers and analysts are often interested in their location and inspection, for example for analysis or maintenance purposes. The proposed approach, by exploiting the formal domain knowledge added to the process model as well as Formal Concept Analysis [64], allows to retrieve candidate crosscutting concerns to be later checked and refined by analysts. In fact, though, when known, crosscutting concerns can be located for example by formulating a BPMN VQL query, when business analysts lack an exhaustive view of their presence, this can be generated by the proposed approach. Moreover, once (manually or semi-automatically) retrieved, crosscutting concerns can be stored (directly or by means of an automated translation) in the form of BPMN VQL queries.

- the opportunity of modularizing, and hence locally managing, crosscutting concerns, by means of a visual aspect-oriented language. The proposed language, BPMN VRL (BPMN Visual Rule Language), is based on rules and extends the BPMN VQL with mechanisms allowing to denote process updates (in detail, additions and removals). BPMN VRL rules allow to describe the crosscutting concern (separately from the main process), as well as its connection points with the process itself. In detail, in this thesis, we focus on the aspect-oriented representation of exception handling mechanisms. In fact, in case of large business processes, in which business designers prefer to focus only on the "happy path", by forgetting the exceptional flows, separate management of exception handling mechanisms is a valid alternative with respect to their complete omission. Again, the graphical syntax of the language, similar to BPMN, aims at relieving BPMN experts from an over-learning effort.

Moreover, in this thesis, we attempt to partially deal with the problems related to the input material required for enabling the automated services (deriving from the use of semantic annotations) described in the previous section, i.e., process models (documenting the application flows) and their semantic annotation with domain ontology concepts.

With respect to the need of recovering process models, when they are not available, we propose a technique that reverse engineers business process models from existing software systems, which expose the business processes by means of Web applications. In detail, the approach: (a) recovers an initial process model from the log files obtained by tracing the exercised User Interface elements of the application; (b) refines the initial model by clustering process elements according to different criteria. It has been shown [149], in fact, that process modularity positively impacts process model understandability and readability, key factors for process models aiming at facilitating the communication among humans and serving as process documentation. The evaluation of the recovered process models, hence, should include an assessment of their understandability, that, however, is not trivial to estimate. In order to overcome this problem, we investigated several process metrics (related to different process properties) to evaluate relevant factors potentially influencing the readability of the recovered processes. The identified metrics are used as early indicators of the quality of the recovered processes.

With respect to the need of semantically annotating process models, we propose an approach for supporting business experts in this activity. In detail, the proposed technique provides designers and analysts with suggestions for the semantic annotation of a process element starting from the analysis of the label of the element itself. The annotation suggestions are computed on the basis of a similarity measure between the text information associated with process element labels and the ontology concepts. In turn, this requires support for the disambiguation of terms appearing in ontology concepts, which admit multiple linguistic senses, and for ontology extension, when the available concepts are insufficient.

Different forms of evaluations (e.g., case studies, performance evaluations) have been conducted for each of the different techniques in order to provide a first assessment of their effectiveness. Moreover, for one technique in particular, a deeper analysis (a study involving human subjects) has been conducted. We performed an empirical study with subjects in order to investigate the effectiveness for business experts (in terms of benefits gained and effort required) of the BPMN VQL with respect to using natural language. In detail, we analysed the ease of understanding and executing BPMN VQL rather than natural language queries on semantically annotated business process models. Our empirical study shows that understanding BPMN VQL queries is easier than understanding natural language ones, thus supporting our proposal of exploiting BPMN VQL queries for documentation purposes. Moreover, the study indicates that formulating BPMN VQL queries is less expensive than executing queries expressed in natural language, thus suggesting that the advantages deriving from the automatic execution of queries on semantically annotated business processes are superior to the difficulties deriving from formulating queries in BPMN VQL.

Though the current study is limited to only one of the possible uses of the proposed approach, we plan, in future works, to investigate the effectiveness of the framework also for the other proposed uses. In detail, we would be interested to execute further empirical studies with humans, so as to corroborate the promising results obtained from our preliminary case studies and evaluations.

1.4 Innovative Aspects

This thesis supports business experts in managing business processes by providing them with techniques for reverse engineering process models, suggesting semantic annotations for process elements, automatically verifying constraints on the process, querying processes, mining and documenting crosscutting concerns, modularizing crosscutting concerns as well as managing exceptions and exception handling. The main contributions of this thesis to the state of the art are hence the following:

1. we propose a novel approach for reverse engineering a process model from the execution traces of a Web application that exposes the process, as well as clustering techniques to be applied on top of reverse engineering for improving process readability[1] [48, 47];

2. we investigate several metrics proposed in the literature for the evaluation of the process model quality and we identified among them the most relevant for the understandability of recovered process models (i.e., the metrics empirically correlated to model understandability), thus innovatively proposing them as early indicators of process model understandability in process mining approaches;

3. we propose the enrichment of BPMN business process models with domain ontology concepts, by means of the semantic annotation of process elements, and the formalization of such information, as well as of process structural information, in a knowledge base [44, 45];

4. we suggest novel techniques to support business designers in the semantic annotation of process elements and in the related domain ontology building/enrichment as well

[1]Though approaches exist in the literature for process modularization (e.g., [4] and [14]), to the best of our knowledge, clustering-based modularization techniques have never been applied on top of process model recovering.

as concept sense disambiguation [50, 51];

5. we propose an innovative approach based on Description Logics for the automated verification of constraints [44, 45, 46, 153];
6. we define BPMN VQL, an easy-to-use, visual language for querying BPMN business processes (and hence also manually retrieving crosscutting concerns in business processes) [49];
7. we investigate a novel technique for semi-automatically mining candidate crosscutting concerns in business processes;
8. we define a visual aspect oriented language for BPMN business process models, allowing to locally manage crosscutting concerns (such as exception handling) and to weave them only when needed;
9. we provide a first evaluation of the advantages of the graphical language BPMN VQL for querying processes.

1.5 Publications

We report in the following the list of workshop, conference and journal papers, in which the material presented in the thesis has been published:

- Ghidini C., Di Francescomarino C., Rospocher M., Serafini L., Tonella P., Semantics based aspect oriented management of exceptional flows in business processes. ((To appear in IEEE Transactions on Systems, Man, and Cybernetics Part C, Special issue on Semantic-enabled Software Engineering.)
- Di Francescomarino, C., Ghidini, C., Rospocher, M., Serafini L., Tonella, P., A Framework for the Collaborative Specification of Semantically Annotated Business Processes. (To appear in Journal of Software Maintenance and Evolution: Research and Practice, 2011).
- Di Francescomarino, C., Marchetto, A., Tonella, P., Cluster-based modularization of processes recovered from web applications. (To appear in Journal of Software Maintenance and Evolution: Research and Practice, 2011).
- Di Francescomarino, C., Tonella, P., Supporting Ontology-Based Semantic Annotation of Business Processes with Automated Suggestions. In: International Journal of Information System Modeling and Design, vol. 1, n. 2, 2010, pp. 59-83.

- Di Francescomarino, C., Ghidini, C., Rospocher, M., Serafini, L., Tonella, P., Semantically-aided business process modeling. In Proc. of the 8th International Semantic Web Conference (ISWC2009), pp. 114- 129.
- Rospocher, M., Di Francescomarino, C., Ghidini, C., Serafini, L., Tonella, P., Collaborative Specification of Semantically Annotated Business Processes. In: Business Process Management Workshops (BPM2009), 3rd International Workshop on Collaborative Business Processes (CBP2009).
- Di Francescomarino, C., Tonella, P., Supporting Ontology-based Semantic Annotation of Business Processes with Automated Suggestions. In: Proc. of the 14th International Conference on Exploring Modeling Methods in Systems Analysis and Design (EMMSAD2009).
- Di Francescomarino, C., Marchetto, A., Tonella, P., Reverse Engineering of Business Processes exposed as Web Applications. In: Proc. of the 2009 European Conference on Software Maintenance and Reengineering (CSMR2009).
- Di Francescomarino, C., Ghidini, C., Rospocher, M., Serafini, L., Tonella, P., Reasoning on semantically annotated processes. In: Proc. of the 6th International Conference on Service-Oriented Computing (ICSOC2008), pp. 132-146.
- Di Francescomarino, C., Tonella, P.: Crosscutting concern documentation by visual query of business processes. In: Business Process Management Workshops (BPM2008), 4th International Workshop on Business Process Design (BPD2008).

1.6 Structure of the Thesis

The structure of the thesis is inspired by the life-cycle of semantically annotated process models, as shown in Figure 1.1. Process models can be either manually designed by business modellers or, in case of legacy systems, they can be reverse engineered starting from the execution logs of the considered application. Once available, the process model can then be enriched with semantic annotations that can be automatically suggested to the annotator. The semantically annotated process model allows to provide business experts with a number of reasoning services, such as the constraint verification, the crosscutting concern retrieval, documentation and aspectization, thus supporting them in process understanding, analysis and management.

In detail, the thesis is structured as described in the following.

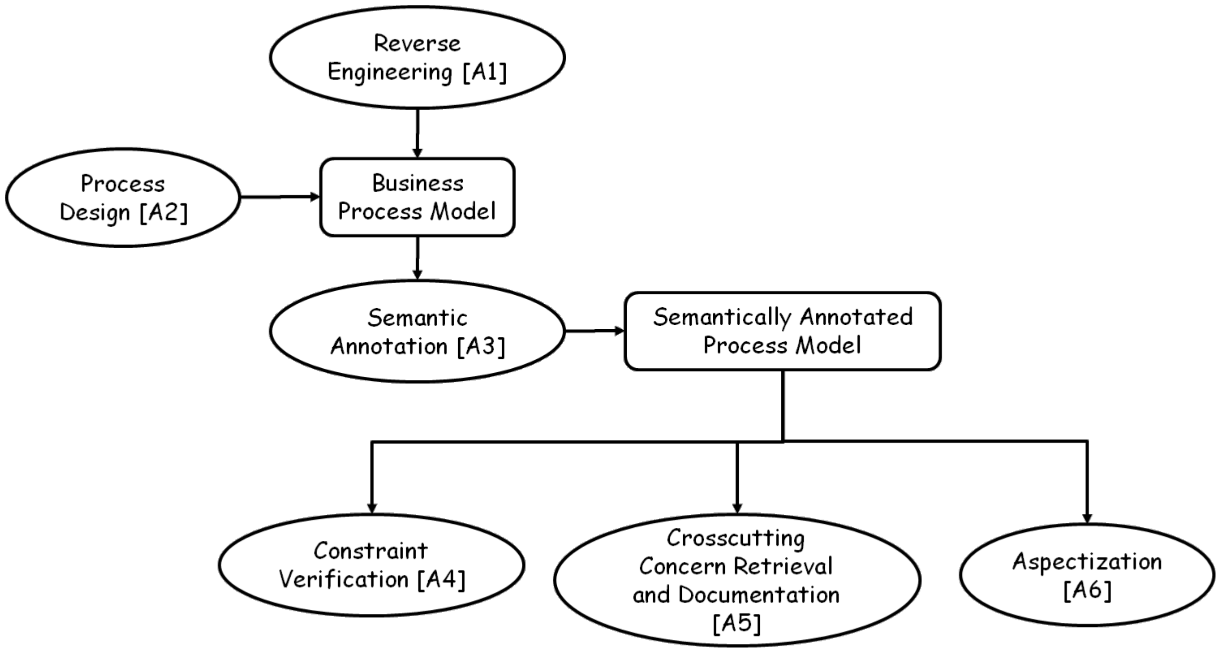

Reverse Engineering [A1]
Process Design [A2]
Business Process Model
Semantic Annotation [A3]
Semantically Annotated Process Model
Constraint Verification [A4]
Crosscutting Concern Retrieval and Documentation [A5]
Aspectization [A6]

Figure 1.1: Thesis flow

Chapter 2 introduces some background concepts used throughout the thesis.

Chapter 3 focuses on cases in which applications are not documented (i.e., no business process is available describing the execution flow). In these situations a possibility is reverse engineering the business processes underlying the applications ([A1] in Figure 1.1). This chapter describes a technique for reverse engineering process models starting from the execution logs. It also presents some approaches for modularizing BPMN process elements into sub-processes in order to limit the complexity of the recovered models. Moreover, this chapter investigates some metrics from the literature for measuring the understandability of recovered process models.

In Chapter 4 we present our proposal related to the semantic annotation of BPMN business processes ([A3] in in Figure 1.1) with concepts belonging to a domain ontology. We exploit the the formalization of process models into a knowledge base. In detail, the elements of a process diagram are annotated with concepts taken from a domain ontology and some techniques are presented for supporting designers in the semantic annotation, by suggesting the candidate annotations .

Chapter 5, Chapter 6 and Chapter 7 show applications of the proposed semantic annotation framework.

Chapter 5 ([A4] in Figure 1.1) demonstrates how semantic annotations can be exploited in order to support business experts to verify structural requirements formalized in Description Logics.

In Chapter 6 ([A5] in Figure 1.1), the same annotations are used for querying the process as well as for retrieving and documenting concerns crosscutting the process.

In Chapter 7 the modularization of crosscutting concerns into aspects is presented. In particular, such a mechanism is proposed as a solution to the problem of exception handling management in business processes, by means of the modularization of exception handlers into aspects. Moreover, exception handling requirements are used to verify the correct management of exception handling.

Aside from case studies and evaluations scattered across the different chapters, Chapter 8 presents a first step towards the empirical verification of one of the proposed approaches. In detail, the advantages provided by the visual language based on BPMN and used for documenting crosscutting concerns are evaluated by means of an empirical study with human subjects.

Chapter 9 presents the main works in the literature related to the research fields in this thesis. It is structured in subsections each related to one of these main topics: reverse engineering approaches for process models and understandability metrics for their evaluation, process model semantic annotations, constraint verification on business processes, works related to the management of crosscutting concerns and aspects in business processes, exceptions and exception handling mechanisms and empirical studies for the evaluation of visual process query languages.

Conclusions are finally presented in Chapter 10.

Chapter 2

Background

In the current chapter we provide an overview of Business Process Management (Section 2.1) and Semantic Business Process Management (Section 2.2), as well as some of the main concepts and notations of these fields used in the remainder of the thesis.

2.1 Business Process Management

Business Process Management (BPM) [162, 175, 191] aims at providing tools and techniques supporting definition, automation and control of business activities for narrowing the gap between business requirements (management intent) and information technologies (management execution) [78, 162]. Business processes, by organizing business activities and managing their integration with people and systems, represent the key instruments for achieving the ambitious goal of bridging business organization and technology [191].

A business process is defined as "a set of activities that are performed in coordination in an organizational and technical environment and that aim at jointly realizing a business goal" [191]. BPM includes concepts, methods and techniques to support the design, management, enactment and analysis of business processes [175]. In other terms, at the core of BPM, there is the explicit representation of business processes with their activities and constraints (Business Process Modelling) that can later on be analysed, improved and enacted [191].

2.1.1 Business Process Models

According to the definition by Weske [191], "A business process model consists of a set of activity models and execution constraints among them". It is, hence, an abstract representation of how a business organization achieves a particular objective by means of a set of business activities. The concrete enactment of this model in the organization's operational business, is instead, realized by activity instances.

According to the four layers of the Meta Object Facility (MOF) specification 1.4.[2], process instances populate the M0 level in process domains; process models, instantiated by process instances, lie in the M1 layer; the process meta-model, instead, is placed in the M2 layer and provide concepts that are associated with the components of the specific notation used for describing the process model [191].

Though a number of languages and notations with different components have been proposed for representing processes and workflows, there exists a subset of components associated with shared concepts in meta-models. A process model usually consists of activities, representing the units of work, connected by a control flow. The control flow is described by means of directed edges as well as of splitting and merging points. Splitting points allow to decide, according to their type and conditions (if any), which outgoing edge to activate, while merging points describe how to reconcile different incoming paths.

Business process notations can be formal, as for example Petri Nets [141], or less formal, as, for instance, EPC (Event-driven Process Chains) [89], YAWL (Yet Another Workflow Language) [174], UML Activity Diagrams [132] and BPMN (Business Process Modeling Notation [25, 131] or Business Process Model And Notation [135, 136]: BPMN acronym has been recently changed from Business Process Modeling Notation to Business Process Model And Notation).

BPMN

Business Process Modeling Notation [25, 131] or Business Process Model And Notation [135, 136] (BPMN) is a (graphical) language for the specification of business processes developed by the Business Process Management Initiative and standardized by the OMG. Though it is a relatively young notation, due to its growing use, four versions have already been released since 2006. The most important changes with respect to the original version [25] have been introduced in the last specifications, BPMN 2.0 [136] specifications, which have been recently released. Hereafter, however, we refer to BPMN v1.1 [131], which is the BPMN version we used throughout the thesis as process model notation.

The main purpose of BPMN is bridging the gap between process models and process implementation [131], thus allowing to represent BPMN process models that are easily readable and usable both by business people and by developers by means of a graphical notation.

The core of BPMN is the specification of a BPD (Business Process Diagram), which is a

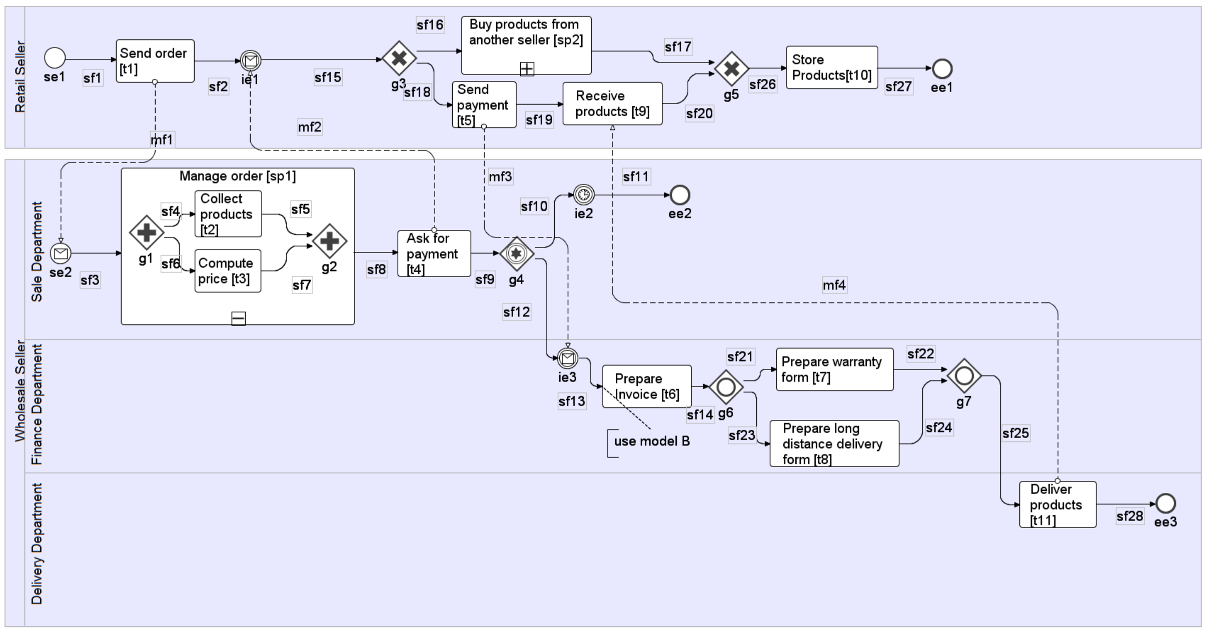


Figure 2.1: An example of Business Process Diagram

diagram designed to be used by the people who model and manage business processes. An example of BPD is shown in Figure 2.1. BPMN also provides a mapping to an execution language of BPM Systems (BPEL4WS) [131].

A BPD is composed of a set of graphical elements. The BPMN graphical elements can be categorized in four main classes:

- The first group is composed of *flow objects*. These are the graphical elements encountered when executing the business process flow and they can be further refined in the following three groups:
  - *events*, which denote something that happens at some time in the process and are represented as circles (e.g., $se1$, $ee1$ and $ie1$ in Figure 2.1);
  - *activities*, which are the real work units in the process, represented as rounded boxes (e.g., $t1$, $t2$ and $sp1$ in Figure 2.1);
  - *gateways*, which represent decision and merging points in a process and are depicted as diamonds (e.g., $g1$, $g3$, $g4$, and $g7$ in Figure 2.1).

  In detail, each of these three groups of flow objects has several specializations. For example, activities can be atomic (*task*, e.g., $t1$ and $t2$) or composed activities (*sub-processes*, e.g., $sp1$ and $sp2$). Events, instead, can be categorized with respect to the type of action they trigger, i.e., the start or the end of a process (*start* and *end events*, e.g., $se1$ and $ee1$, respectively), or the continuation of the process flow execution (*intermediate events*, e.g., $ie1$ and $ie2$), but also with respect to the type of event (e.g., *message event* as $ie1$ or *timer event* as $ie2$). Finally, gateways can be classified with respect to their role of decision or merging points, but also with respect to the type of semantics the specific decision or merging point conveys (e.g., in case of *AND* decision gateways, as $g1$, all the outgoing flows have to start in parallel; in case of *OR* decision gateways, as $g6$, at least one of the outgoing flows has to start; and in case of *XOR* gateways, both *data-based* as $g3$ or *event-based* as $g4$, only one of the outgoing flows has to start).

- Graphical objects that allow to connect flow objects populate the second group. They can describe the process control flow (*sequence flows*, which are depicted as arrows between pairs of flow objects, as $sf1$ and $sf2$ in Figure 2.1), the communication between flows of the process realized by different participants (*message flows*, which are represented as dashed arrows, as $mf1$ and $mf2$ in Figure 2.1), and, finally, the association of additional information to flow objects (*associations*, which are depicted as dashed lines, as the dashed line connecting the task $t6$ to the annotation *usemodelB* in Figure 2.1).

- The third group of graphical elements is composed of *swimlanes*, objects aiming at clarifying the process organization. They can be either *pools* (e.g., *RetailSeller* and *WholesaleSeller* in Figure 2.1), representing participants, e.g., in B2B (Business To Business) contexts, and *lanes* (e.g., *SaleDepartment* and *DeliveryDepartment* in Figure 2.1), sub-partitions of pools that are often used for representing roles, in that they allow to organize elements and categories.

- Artefacts, finally, populate the fourth group. They allow to represent additional information that does not affect the process control and interaction flow. An artefact, in turn, can be: a *text annotation*, as *usemodelB* in Figure 2.1, allowing to specify additional text information (it can be connected to a specific graphical object by means of an association); a *data object*, used for modelling documents and data exchanged as input and output between activities (it is depicted as a box with a folded corner); and *groups* for clustering flow objects according to logical categories (they are depicted as dashed boxes grouping the graphical objects to be clustered in the same category).

2.2 Semantic Business Process Management

Semantic Business Process Management (SBPM) [56, 78] originates from the need of providing an automated way for querying business processes and reasoning about them. To this purpose, Hepp et al. [78] in their visionary paper, advocate for the need of providing a machine-readable representation of business processes at a semantic level and propose the integration of semantic technologies (ontologies and Semantic Web Services) with business process management.

Each phase of the business process life-cycle (modelling, implementation, execution and analysis) is affected by this new semantic dimension, as well as potentially enhanced with increased automation and extended functionalities [56]. In detail, Wetzstein et al. [56] envisage the following benefits:

- discovery and auto-completion of process fragments in the process modelling phase (subject to process model semantic annotation);
- Web service discovery and composition in the implementation phase, by exploiting the semantic annotation of the modelling phase;
- run-time discovery (e.g., in case of new information available only at run-time) in the execution phase;
- process monitoring, mining, querying and analysis by exploiting reasoning on semantically annotated logs and models;

2.2.1 Semantic Annotation of Process Models

According to Wetzstein et al. [56], semantic annotation represents the first step for enabling querying and reasoning services in business process life-cycle. Semantic annotations can be added to documents, web pages, models, in order to define or clarify their hidden semantics. To this purpose, semantic annotation takes often advantage of ontologies (whose objective is to provide a shared, formal and explicit specification of concepts), thus resulting in an approach associating ontology concepts to the information being annotated. According to Lautenbacher et al. [98], also in case of process models, it is possible to characterize semantic annotation according to the level at which it is performed, similarly to the distinction between document-level (i.e., concerning the whole document) and character-level (i.e., concerning only specific parts of the document) annotation in linguistics. In detail, they identify a *metamodel-level annotation* and a *model-level annotation.* Metamodel-level annotation occurs when constructs of a process model are annotated (e.g., with ontologies describing process model notations, as sBPEL [129] or sBPMN [3]), while model-level annotation consists of annotating process model elements (e.g., with domain ontologies).

2.2.2 Ontologies

An *ontology* is a "formal, explicit specification of a shared conceptualisation" [74]. It allows to describe in a formal way a domain by specifying its concepts and the relations between concepts. However, two essential characteristics distinguish ontologies from other types of models [57]:

- they provide a formal semantics, thus allowing machines to reason about them;
- they define real-world semantics, so that, based on a consensual terminology, they allow also humans' comprehension.

These two characteristics make ontologies suitable for three main purposes [76]: (i) the communication among humans, among machines and among humans and machines; (ii) exploitation of reasoning; and (iii) knowledge reuse.

Differently from controlled vocabularies, taxonomies and thesauri, which only allow a restricted description of their terms/concepts, ontologies allow to express customized relationships between concepts. From a structural point of view, in fact, an ontology is usually composed of concepts, also called classes, individuals (or class instances), relations between concepts and attributes, i.e., class properties. From the content viewpoint, instead, ontologies can be classified according to their level of abstraction, into *upper-level* and *domain* ontologies. An upper-level ontology (*top ontology*) is an ontology describing very general concepts (as for example, `object` and `action`), spanning across and abstracting over different domains. SUMO[1], DOLCE [63] and GFO [79] represent some examples of widely recognized upper level ontologies. A domain ontology (or domain-specific ontology) models a specific domain (e.g., the On-line shop domain) and uses specific concepts (e.g., `cart` and `checkout`) for its description.

Ontologies can be represented by using specific languages: CycL [99] and OWL [134] are examples of this type of languages.

OWL

Web Ontology Language (OWL) is a family of languages for representing ontologies. It is an extension of RDF (Resource Data Framework) and it aims at enabling reasoning by

[1]`http://protege.stanford.edu/ontologies/sumoOntology/sumo_ontology.html`

providing a formal and machine-interpretable semantics. Two versions of OWL have been standardized: OWL [134] and OWL 2 [133], each including, in turn, variants of different expressiveness.

In detail, OWL specifications [134] propose three variants of increasing expressiveness (i.e., each of them is an extension of the previous sub-language) of the OWL language: OWL-Lite, OWL-DL and OWL-Full. OWL-Lite was conceived as a light version of OWL with a low expressiveness ($\mathcal{SHIF}(\mathcal{D})$) aimed at supporting the hierarchical classification and simple properties. OWL-DL is based on a strict correspondence with Description Logic. This allows to maximize the expressiveness (it includes all OWL constructs with an expressiveness of $\mathcal{SHOIN}(\mathcal{D})$ such that only specific combinations of restrictions are not allowed), while preserving the advantages of being formally grounded to Description Logic (e.g., the availability of practical algorithms with known complexity). Description Logics (DLs), in fact, are a family of logics that are decidable fragments of the First Order Logic (FOL) with desirable properties. Both OWL-Lite and OWL-DL are based on DL, while OWL-Full was mainly conceived for compliance with RDF, that it semantically extends. It enriches FOL with new constructs but it is undecidable. OWL 2 DL has an expressiveness of $\mathcal{SROIQ}(\mathcal{D})$.

OWL allows to describe *classes* (corresponding to DL *concepts*), e.g., Person. Classes can be organized hierarchically by exploiting the is_a relationship (corresponding to the DL subsumption, $\sqsubseteq$), so that sub-classes inherit super-class properties. For example a class Child is a sub-class of the class Person. The root class of this hierarchy is the `owl:Thing` class (DL *top*, $\top$), while the common leaf is the `owl:Nothing` (DL *bottom*, $\perp$). Moreover, OWL allows the representation of *instances* and of *properties*. Instances correspond to DL *individuals*, while properties, corresponding to DL *roles*, are oriented relationships between two objects and, as such, allow to specify a domain and a range. For example Mark is an individual of the class Person, while has_person_pet is a property having as domain the class Person and as range the class Pet. In detail, properties can be of two types: *object properties* between instances of two classes (e.g., has_person_pet) and *datatype properties* between class instances and literal data (e.g., is_person_age_old, where age is of type integer). Moreover, each property can be defined as symmetrical (in case the relationship is bidirectional), transitive (i.e., if $p(x, y)$ and $p(y, z)$, then $p(x, z)$, where p is the property and x, y and z individuals) or functional (i.e., for each x in the domain, only one y in the range is allowed) and a property can be defined as the inverse of another property (i.e., if $p(x, y)$, the inverse of p, p^{-1} is such that $p^{-1}(y, x)$). For example, the object property is_person_married_with (having as both domain and range the class Person), is symmetric; an example of transitive property is the object property is_elder_than; a functional property could be the datatype property is_person_age_old; while the object property is_wife_of is the inverse of the property is_husband_of. Finally, several operators like union, intersection, complement, enumeration, disjointness, quantifiers, cardinality can be used for defining special classes as composition of others, as well as to specify special properties among classes (e.g., disjointness).

OWL is supported by many available reasoners, as for example, Pellet[2], RacerPro[3], Fact++[4].

[2]http://clarkparsia.com/pellet/
[3]http://www.racer-systems.com/
[4]http://code.google.com/p/factplusplus/

Chapter 3

Reverse Engineering of Business Process Models

"All truths are easy to understand once they are discovered; the point is to discover them."
Galileo Galilei

Business processes operated by companies are often scarcely documented and, even when some form of documentation exists, this is often incomplete and inconsistent with the actual realization. Nevertheless, accurate and consistent business process documentation is crucial to have a clear understanding of the business flow and to make strategic business decisions (e.g., aimed at maintaining or restructuring existing processes).

Many business processes in companies are realized as software systems and Web applications are often the preferred way of exposing them to the users. In fact, Web applications can be implemented on a wide variety of platforms and in many different environments provided a browser and an Internet connection are available.

In such a scenario, an appealing option is reverse engineering the documentation about the business processes from the implementation, so as to ensure that it exactly represents the actual process, as realized in the software system supporting it. Reverse engineering is an interesting option even when some form of documentation exists, since this can be improved and aligned with the actual implementation [126].

Several works in the literature propose techniques to recover process models starting from the static analysis of different artefacts, such as the software specification or the source code (e.g., [197], [67], [138]) or by dynamically investigating the application execution, for example by analysing the execution logs (e.g., [24], [179], [178]). In all cases, the resulting process should be reasonably close to the actual application while preserving a good degree of understandability[1]. Static techniques, though allowing to better control the process complexity, have limited adherence to the process execution. On the other hand, one of the main problems with techniques extracting processes from execution logs is that the recovered models are often large, complex and intricate, thus resulting difficult to read and understand (the so called "spaghetti" processes [185]). The assessment of the understandability of recovered models plays hence a key role in the evaluation of the quality of recovered process models. This is, however, not trivial to estimate, lacking an overall and shared means to measure it.

In this chapter we propose a dynamic technique that recovers business processes from application execution traces, while exploiting modularization techniques for limiting the model complexity and hence easing their readability (Section 3.1). Empirical evidence shows in fact that modularity affects positively process understandability [149]. In detail, exploiting the role of Web Applications as a means for exposing processes to the users, we focus our analysis on exercised Web-GUI elements captured by tracing the application execution with the aim of extracting initial process models. The initial processes are then refined with different clustering techniques in order to improve their understandability. Moreover, in the perspective of measuring the quality of the recovered process models, we investigate and empirically evaluate a set of metrics proposed in the literature to asses the quality of processes in terms of understandability and readability (Section 3.2).

The material presented in this chapter related to the reverse engineering of business processes from execution traces and their clusterization has been published in [48, 47].

3.1 Reverse Engineering of Business Processes

In this section we present a technique for the reverse engineering of BPMN business processes from Web applications. Differently from reverse engineering approaches that statically infer process models from software artefacts, our technique analyses the application dynamically. It is often the case, in fact, that software artefacts are not fully available (e.g., in case of third-party component code). Moreover, Web applications are intrinsically dynamic, (e.g., pages can be constructed dynamically, the DOM structure is dynamically manipulated by means of reflection mechanisms), which make them hard to be statically analysed. Finally, Web applications are based on user events, i.e., their operations are mainly guided by user actions, which cannot be captured statically.

In detail, the approach we propose analyses application execution logs, similarly to

[1]According to the related literature (e.g., [154] and [185]), we refer to model understandability in terms of the extent to which the artefact (e.g., a software, a model) is easily comprehended with respect to purpose and structure ([20]).

process mining techniques (e.g., [24], [179]), which try to infer processes by analysing the workflow logs, i.e., traces containing real and traced information about organizations and process executions. However, in our case, execution logs are not provided automatically by the execution environment, hence a preliminary step is required in order (to select and) trace the information of interest. Since Web applications are mainly guided by actions performed by users, our technique focuses on the analysis of the user interactions with the Web application, which are made available from the application Web-GUI. Web application forms and links implicitly contain information about the underlying process and its elements [90]. In fact, accomplishing a user task usually involves providing (through forms and links) some data and requesting the server to perform operations (server-side tasks). The idea is hence to trace the application execution for capturing information about the exercised GUI elements and using it to infer an (initial) process model, as exposed on the Web.

Furthermore, since the initial recovered process can be very detailed, flat and large, different clustering techniques are applied to modularize and organize the process, as well as to increase its readability and understandability [149].

In the next subsection (Subsection 3.1.1), we provide some concepts used in the remainder of the chapter about modularization and flow graph analysis. Subsection 3.1.2 introduces the business process recovery technique, while Subsection 3.1.3, Subsection 3.1.4 and Subsection 3.1.5 detail its main steps. Finally Subsection 3.1.6 presents a tool supporting the presented reverse engineering technique and Subsection 3.1.7 a case study for the evaluation of the approach.

3.1.1 Background

Modularization

Modularity is the design principle of having a complex system composed of smaller subsystems that can be managed independently though functioning together as a whole. Similarly, in process modelling, modularity is the design principle of having a complex business process composed of smaller sub-processes.

Modularization in process models [149, 111] is mainly a modelling style (i.e., stepwise task refinement) but it also helps in stimulating model reuse and concurrent execution (sub-processes may be executed by means of different engines). In addition, modularization helps programmers and designers in model and software understanding. In a recent experiment related to process modularization, Reijers et al. [149] showed that process modularization impacts positively process understanding.

However, drawbacks can limit this impact. For instance, no objective criteria and precise guidelines exist to identify the adequate process modularization granularity as well as no unique way exists to modularize a process model. As shown by Reijers et al., alternative process modularization may affect differently the ease of process understanding.

Flow Analysis

Static code analysis is often realized by extracting the control flow graph (CFG) and propagating flow information inside it. In the CFG, each node represents a statement and an edge between two statements represents a (syntactically) possible execution flow between them. For example, the CFG of the fragment of code in Figure 3.2a is depicted in Figure 3.2b (see below for details). The label of each node of the graph is the line number of the corresponding code statement. Properties holding for a certain CFG (e.g., data and control dependences) can be determined by traversing the graph and propagating proper information. The procedure performed for the propagation (through the CFG) of the flow information and its modification according to statement computations can be described in a general way by the Flow Analysis Framework [5]. Given a graph $G = (N, E)$ where N is the set of nodes and E the set of edges, the Flow Analysis Framework consists of:

1. a set V of values that can be propagated in the graph. Elements of V are assigned to *IN[n]* (input of n) and *OUT[n]* (output of n) for each node n in the graph.

2. a transfer function f (or a set of them). Every node n is associated to a function $f_n : V \to V$ representing the computation performed in the node: $OUT[n] = f_n(IN[n])$. The transfer function is assumed to be monotonic.

3. a confluence (*meet*) operator $\bigwedge$ used to join flow values coming from the OUT set of the predecessor (or successor) nodes into the IN set of the current one n: $IN[n] = \bigwedge_{m \in z(n)} OUT[m]$, where m is a node of the node set *z(n)* defined according to the direction of the information propagation (see below). Each meet operator has an identity (*top*) element $\top$. Examples of meet operators are union and intersection operators, respectively having $\top = \emptyset$ and $\top = U$, when $V = 2^U$. The meet operator is associative, commutative and idempotent.

4. the direction of the information propagation can be *forward* or *backward*. This determines if the information is propagated starting from predecessors or successors of a node. In case of forward propagation, the meet operator is applied to the set of

```
** Initialization **
For each $n \in N$
 $IN[n] = \top$
 $OUT[n] = f(IN[n])$
End For

** Flow propagation **
While any $OUT[n]$ or $IN[n]$ changes
 For each $n \in N$
  $IN[n] = \bigwedge_{m \in z(n)} OUT[m]$
  $OUT[n] = f_n(IN[n])$
 End For
End While

where:
 if (direction = ``fwd'') $z = pred$
 if (direction = ``bwd'') $z = succ$
```

Figure 3.1: Flow Analysis Framework

predecessors of n ($z(n) = pred(n)$), while in the other case it is applied to the set of successors of n ($z(n) = succ(n)$).

The generic flow analysis algorithm is reported in Figure 3.1. The algorithm starts with an initialization step in which IN and OUT sets are defined for each node of the graph. Then the information is propagated in the graph by applying meet operator and transfer function, according to the selected propagation direction, until the fix point is reached.

The flow analysis framework has been successfully applied for determining different properties of a program, e.g., reachable uses, dominators, postdominators. For each type of analysis, the framework is instantiated by defining the most suitable: (1) set of flow values V, (2) transfer function f, (3) meet operator $\bigwedge$, and (4) propagation direction.

For example, the instantiation of the flow analysis framework for the computation of the dominators of the nodes in a CFG will propagate the information related to the nodes themselves. A node n in a CFG, in fact, dominates a node m in the same graph if n is contained in every path of the CFG from the initial node of the CFG to m. The flow analysis framework, in this case, can be instantiated in the following way:

1. V contains sets of nodes of the CFG (i.e., $V = 2^N$, where N is the set of graph nodes).

2. The transfer function f has the following structure:

$$f(x) = GEN[n] \cup (x - KILL[n])$$

where $GEN[n]$ and $KILL[n]$ are defined as:

$$\begin{aligned} GEN[n] &= \{n \in N\} \\ KILL[n] &= \emptyset \end{aligned}$$

Hence, for each iteration of the flow propagation, $OUT[n] = GEN[n] \cup IN[n]$.

3. The confluence (*meet*) operator is set intersection, thus detecting, for each node n, only those nodes that have to be necessarily traversed for reaching node n itself. The corresponding top element is $\top = N$.

4. The direction of the information propagation is *forward*, thus the meet operator is applied, for each node n, to the set of predecessors of n ($z(n) = pred(n)$).

The graph in Figure 3.2b shows next to each node the corresponding IN and OUT values both in the initialization phase (IN_0 and OUT_0) and in the only iteration of the algorithm occurring in this example (IN_1 and OUT_1). Next to each statement in Figure 3.2a is reported the final set of statements dominating the current statement. For instance, the code statement at line 3 ("goto a;") is dominated by the statements at lines 1 and 2 (respectively: "x=3;" and "if (x)").

3.1.2 GUI-based Reverse Engineering

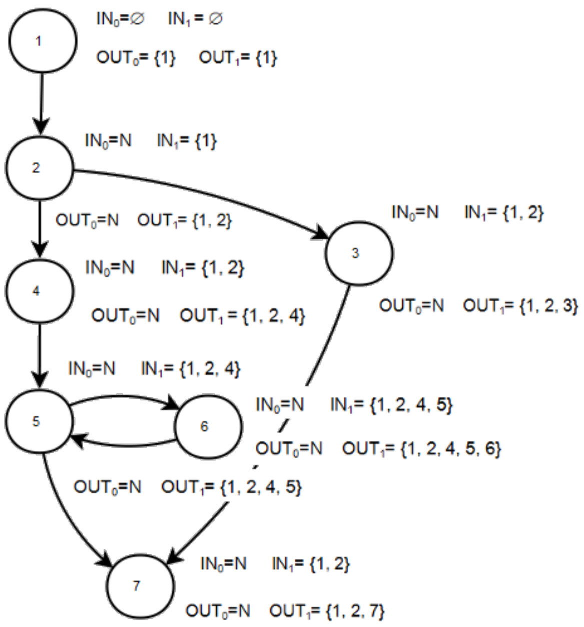
The proposed reverse engineering approach for extracting the process underlying a Web application is based on the analysis of the application GUI exercised during the execution.

Figure 3.3 shows the overall approach. It is performed by means of the following main steps: (A) trace of application execution; (B) process extraction from the recorded traces; and (C) process refinement by means of clustering. The idea is to build an initial process that describes the workflow exposed via the application GUI by analysing execution traces. The traces contain information about the behaviour of the application (in terms of visited pages and executed events, such as button clicks and links), the structure of each visited page (in terms of its forms and links) and its content (in terms of textual content related to GUI-elements). By analysing such traces the initial process is inferred and represented in BPMN. Afterwards, different clustering techniques (i.e., D. structural, E. page-based, F. dependency-based and G. semantic) can be applied to the resulting process for its refinement and for improving its modularization. As shown in Figure 3.3, different combinations of clustering techniques can be applied to the recovered process (e.g., structural and dependency-based clustering, only structural clustering). However, different clustering combinations and orders can impact process modularization in different ways.

```
Code statements   Dominators

main()
{
1.   x=3;           {1}
2.   if (x)         {1, 2}
3.    goto a;       {1, 2, 3}
4.   y=x+1;         {1, 2, 4}
5.   while(y)       {1, 2, 4, 5}
6.    y--;          {1, 2, 4, 5, 6}
7.   a:...          {1, 2, 7}
}
```

(a)


1
$IN_0=\varnothing$ $IN_1=\varnothing$
$OUT_0=\{1\}$ $OUT_1=\{1\}$
2
$IN_0=N$ $IN_1=\{1\}$
$OUT_0=N$ $OUT_1=\{1, 2\}$
3
$IN_0=N$ $IN_1=\{1, 2\}$
$OUT_0=N$ $OUT_1=\{1, 2, 3\}$
4
$IN_0=N$ $IN_1=\{1, 2\}$
$OUT_0=N$ $OUT_1=\{1, 2, 4\}$
5
$IN_0=N$ $IN_1=\{1, 2, 4\}$
$OUT_0=N$ $OUT_1=\{1, 2, 4, 5\}$
6
$IN_0=N$ $IN_1=\{1, 2, 4, 5\}$
$OUT_0=N$ $OUT_1=\{1, 2, 4, 5, 6\}$
7
$IN_0=N$ $IN_1=\{1, 2\}$
$OUT_0=N$ $OUT_1=\{1, 2, 7\}$

(b)

Figure 3.2: Example of application of the Flow Analysis Framework for the computation of dominators in a code fragment

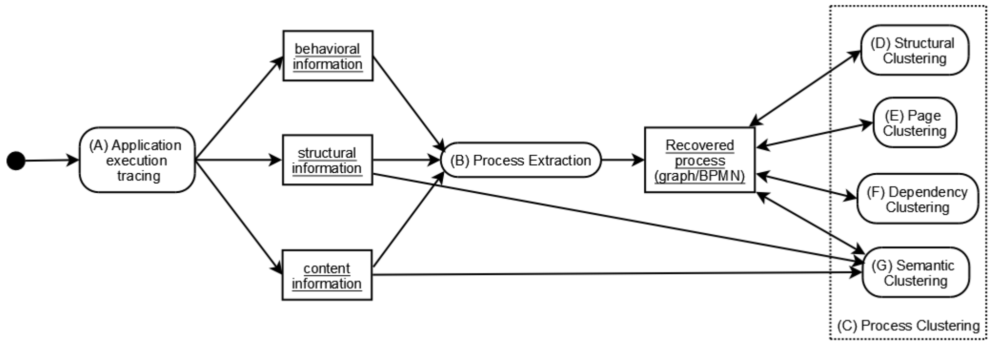


Figure 3.3: Overall picture of the business process reverse engineering approach. Ovals represent activities while squares the produced artefacts.

3.1.3 Dynamic Process Extraction

During the application execution, three types of information are stored in the execution traces:

- *structural information*: the structure of each visited client Web page in terms of all its links, forms and their fields, e.g., input text and checkbox;
- *behavioural information*: the sequence of GUI-elements (forms and links) activated by the user during the navigation of the application and used to submit information/requests to server-side components;
- *content information*: the (textual) content of the application Web pages related to each GUI element part of structural or behavioural information (this information will be used only in the process clustering phase C).

Let us consider for example the user navigation described in Figure 3.4 for the user login. If the user is already registered, he only needs to insert his login and password in the *LoginForm* form and to sign-in, by pressing the *Sign-in* button. However, in our example, he is not yet registered, thus he needs to click the *RegisterNow* submit button. Such an event triggers the *RegisterNow.do* server component (e.g., a servlet), which displays the *RegistrationPage* page. Now the user can choose whether to proceed with the short or complete registration. In this example, the *RegistrationForm* is filled and its *Submit* button pressed by the user. Hence, the *Registration.do* server component is executed and the *ConfirmationPage* is visualized. For this example of navigation we store both behavioural information (e.g., the pair ⟨*RegisterNow*, *RegisterNow.do*⟩) and structural information (e.g., the *Login* and *Password* fields in the *LoginForm* of the *WelcomePage*).

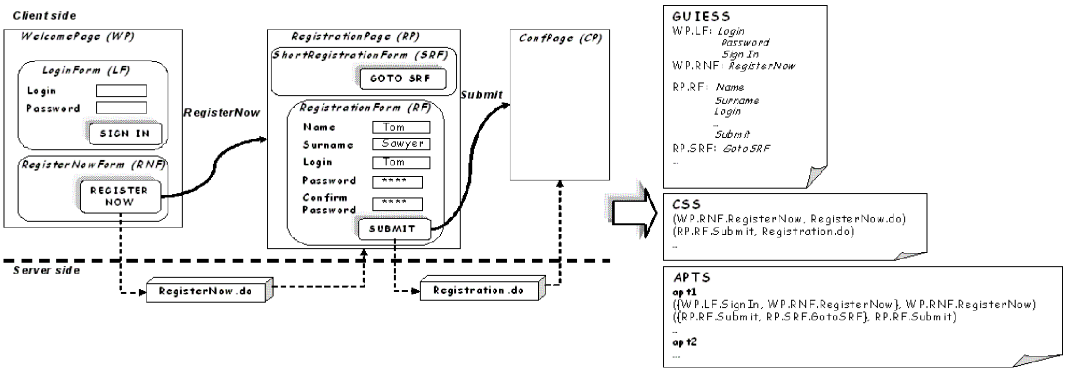


Figure 3.4: An example of user navigation trough web pages

Starting from the set of traces t obtained during the application execution, the initial workflow process is built by applying the algorithm in Figure 3.5.

During the initialization (step 1 of the algorithm), execution traces are analysed for initializing three sets, $GUIESS, CSS, APTS$. Figure 3.4 shows the three sets built on the basis of our example. $GUIESS$ contains the structural information related to each visited Web page in terms of its forms, fields and links (e.g., in Figure 3.4 $GUIESS$ contains information related to both the *LoginForm* and *RegisterNowForm* forms of the page *WelcomePage*). CSS contains the behavioural information related to pairs of submission events and the associated triggered server components (e.g., in Figure 3.4 CSS describes the connection between form submission events, such as *RegisterNow* in the form *RegisterNowForm*, and their server-side components, such as *RegisterNow.do*). Finally, $APTS$ is a set of pair-action traces, obtained by exploiting both structural and dynamic information. $APTS$ is structured in sequences of action pairs ap such as: $ap = (WP, se)$, where WP is a Web page abstraction and se is the form/link submission event, contained in WP and exercised by the user (e.g., in $APTS$ in Figure 3.4, the first action pair trace $apt1$ contains the beginning of the sequence of exercised events, i.e., *RegisterNow* and *Submit*, each associated to the Web page abstraction in which it is contained, i.e., *WelcomePage* and *RegistrationPage*, respectively).

By analysing the action-pair traces we build a finite state machine (FSM, step 2) then used for building the client side process p. The FSM is realized (line 2.1) by defining a node for each GUI-element of the traced Web pages (e.g., *LoginForm* and *RegisterNowForm* in the example). For each pair of consecutive ap (ap_i and ap_{i+1}), an edge is added to the FSM (line 2.2) from the node representing the source form/link in ap_i to all forms/links contained in the target page of ap_{i+1}. For instance, in $APTS$ of the example the pairs $(\{SignIn, RegisterNow\}, RegisterNow)$ and $(\{Submit, GotoSRF\}, Submit)$

```
Input: Set of traces($t$)
Output: BPMN model

1. Initialize
1.1 extract the GUI-element structure set $GUIESS$
    $GUIESS = \{ges \mid ges = (ge,\ structuralComponents(ge))\}$
     where $ge$ is a a form or link GUI-element in $t$ and
      $structuralComponents(ge) = \begin{cases} fields(f) & if\ ge\ is\ a\ form\ f \\ \{l\} & if\ ge\ is\ a\ link\ l \end{cases}$
     and $fields(f)$ is the field set of a form $f$
1.2 extract the set of client-server component pairs $CSS$
    $CSS = \{cs \mid cs = (se,\ ss)\}$
     where $se$ is the submit event exercised by the user via submit button or link
     and $ss$ the correspondent triggered server component
1.3 extract the set of pair-action traces $APTS$
    $APTS = \{apt \mid apt = \langle ap_1, \ldots, ap_z\rangle\}$
     where $ap_i$ = ($WP_i$, $se_i$)
     $WP_i$ = $\bigcup_{ge \in GUIElements(WP_i)} submitElement(ge)$
     and $submitElement(ge)$ is the submit event (i.e. the submit button or the link)
     of the GUI-element (respectively the form or the link) exercised by the user
2. Build client-side $FSM = (N,\ E)$
2.1 $N$ = $\bigcup_{apt\ \in\ APTS} \bigcup_{ap\ =\ (WP,\ se)\ \in\ apt} GUIElements(WP)$
2.2 $E$ = $\bigcup_{apt=\langle\ldots,\ (WP_i,\ se_i),\ (WP_{i+1},\ se_{i+1}),\ \ldots\rangle \in APTS} \bigcup_{se_j \in WP_{i+1}} (GUIElement(se_i), GUIElement(se_j))$
     where $GUIElement(se)$ is the GUI-element (i.e., form or  link) activated by exercising the submit event
     (i.e. the submit button or the link respectively)
3. Convert client-side FSM to the client BPMN process $p = (cPool,\ MF)$
3.1 create the client pool $cPool = (T,\ S,\ G,\ E,\ SF)$
3.2 add tasks and sequence flows
3.2.1 for each $n$ in $N$: add $t_n$ to $T$
3.2.2 for each ($n_i$, $n_j$) in $E$: add ($t_{n_i}$, $t_{n_j}$) to SF
3.3 add necessary gateways to $G$
3.4 populate subprocesses with structural information
      for each $ges = (ge, structuralComponents(ge))$ in $GUIESS$
        if $|structuralComponents(ge)| > 1$ (i.e. if $ge = f\ \&\&\ |fields(f)| > 1$)
3.4.1     move $t_{ge}$ from $T$ to $S$
3.4.2     for each $field$ in $fields(ge)$
            add $t_{field}$ task to $t_{ge}$
4. Add server-side to the BPMN process $ps = (cPool,\ sPool,\ MF)$
4.1 add the server pool $sPool = (sT,\ sS,\ sG,\ sE,\ sSF)$
4.2 create server tasks, events and process message flows
      for each $cs = (se,\ ss)$ in $CSS$
4.2.1   add $t_{ss}$ to $sT$,
        $e_{ss}$ to $sE$ and
        ($e_{ss}$, $t_{ss}$) to $E$
        if $structuralComponents(GUIElement(se))$ = $\{se\}$
4.2.2     add ($t_{GUIElement(se)}$, $e_{ss}$) to $MF$
        else
4.2.3     add ($t_{se}$, $e_{ss}$) to $MF$
4.3 add necessary gateways to $sG$
```

Figure 3.5: Business process recovery algorithm

are two consecutive action pairs representing the fact that the event *RegisterNow* of the form *RegisterNowForm* in the page *WelcomePage* connects the *WelcomePage*, represented as the sequence of the submit events of its forms ($\{SignIn, RegisterNow\}$), to the page *RegistrationPage*, represented as $\{Submit, GotoSRF\}$. Hence, an edge is added to the FSM from the node *RegisterNowForm* to all the nodes in the *RegistrationPage* (i.e., *ShortRegistrationForm* and *RegistrationForm*).

The built FSM is converted into a BPMN process. Figure 3.6 shows a fragment of the BPMN process obtained by applying the algorithm to the example. An empty process is initially created and a client pool is added to it (line 3.1), Figure 3.6 top. For each node of the FSM, a task (line 3.2.1) and the set of sequence flows (line 3.2.2) corresponding to the FSM-edges are added to the pool. For instance, in Figure 3.6, the tasks *RegisterNow* (corresponding to the node *RegisterNowForm*), *GotoSRF* (corresponding to the node *ShortRegistrationForm*) and *RegistrationForm*, as well as the sequence flows corresponding to the edges connecting the first task with the second and the third ones, are added to the client pool. Moreover, the required BPMN gateways are added in decision and merging points (line 3.3), e.g., for tasks with more than one incoming or outgoing edge. The process p is enriched by considering also the structural information in $GUIESS$ (line 3.4). If a task of the process represents a form of the application GUI, the task is converted into a sub-process (line 3.4.1) populated with a set of tasks, one for each field of the form (line 3.4.2). For instance, the form *RegistrationForm* of the page *RegistrationPage* contains several fields; each of them corresponds to a task in the related sub-process *RegistrationForm* in Figure 3.6.

Finally, the server component activation and execution are modelled in the process. To this aim, a server pool is added (line 4.1), Figure 3.6 down. The pool is then populated (line 4.2.1) with events and tasks representing respectively server events and components (e.g., servlets) triggered by the user. Moreover, by analysing each pair (se, ss) in CSS, gateways, sequence and message flows are added to the process. In particular, a message flow is created between the two pools by connecting each se task contained in $GUIElement(se)$ with its server-side counterpart (the event firing the activated servlet). For instance, in the example in Figure 3.6, the server-side component *RegisterNow.do*, called by the form *RegisterNowForm*, represents a task of the server pool activated by an event triggered by the client pool task *RegisterNow*.

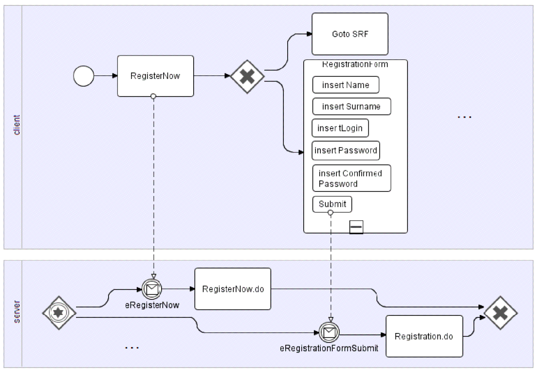


Figure 3.6: Fragment of the process example

3.1.4 Process Clustering

The client pool of the recovered process may be large and complex. We improve its readability by modularizing it [149]. In detail, sub-processes, grouping cohesive and meaningful sets of process tasks, are identified and introduced in the process by applying clustering techniques.

We consider the following clustering criteria: *structural clustering* (the structure of each group of nodes in the graph matches a loop, a sequence or an alternative pattern); *page clustering* (all the nodes in each group represent elements of the same Web page); *dependency clustering* (each group of nodes minimizes *coupling* with other groups and maximizes internal *cohesion*); and *semantic clustering* (the nodes in each group correspond to page elements with similar textual content).

These techniques can be applied individually or composed in specific orders for improving the process modularization. The input to each clustering technique is a graph extracted from the recovered process (a graph node corresponds to a process task and a graph edge to a sequence flow) and used as intermediate artefact for grouping process elements. The produced output is a refined BPMN process, whose client pool (obtained by converting the clustered graph back to a new BPMN process) contains a set of sub-processes which replace the clustered process elements. Furthermore, in order to improve process understandability, for each sub-process some characterizing terms are identified by analysing the terms contained in such a sub-process. These terms are suggested to the analyst as meaningful sub-process labels.

In the rest of this section we present each clustering technique.

Structural Clustering

This type of clustering was inspired by structured programming [40]. Several structuring techniques and methodologies have been proposed over time for removing or reducing the use of GOTO statements (i.e., writing so called "structured programs"). Structured programs are composed of simple (single-input, single-output), composable and hierarchical program-flow structures, such as sequence, selection, and repetition.

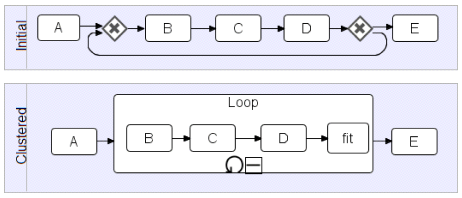
By means of structural clustering we structure the process by grouping a set of graph nodes (process tasks) if their arrangement matches a structural pattern such as loop, sequence and alternative pattern. Each structural pattern is detailed in this section. We identify "blocks" of nodes representing tasks in the initial process and group them into sub-processes. A set of graph nodes (process tasks) is a *block* if: (i) it respects at least one of the three clustering criteria (see below); and (ii) no nodes have edges incoming into or outgoing from the block (with source or destination not in the block), except for exactly one source and one sink node. This notion of block has already been used in some existing works (e.g., [137] and [182], in the process conversion from BPMN to BPEL).

Loop Clustering

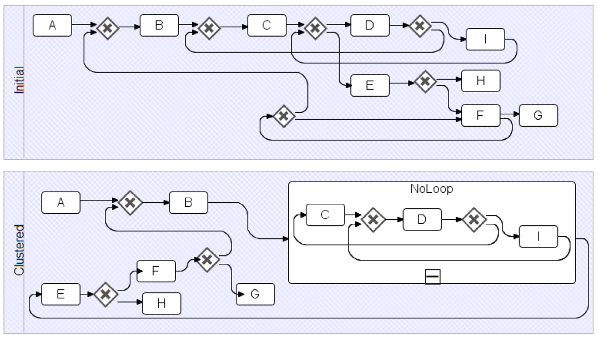
Loop clustering groups tasks whose flow creates a cyclic structure. Such a pattern is recognized by applying the algorithm by Tarjan [166] for computing the strongly connected components (SCCs) in the process. The identified SCCs are then filtered by considering only those of them having at most one node with incoming SCC-external edges (i.e., edges whose extreme nodes are not both contained in the SCC) and at most one node with SCC-external outgoing edges (the target node is out of the SCC), respectively representing the SCC source and sink nodes.

After filtering, each remaining SCC is mapped to a cyclic sub-process of the initial process. If the SCC, instead, contains an edge from the sink to the source node (*return edge*), the SCC is transformed into a BPMN loop sub-process, in which a fictitious exit node is added, the target of all the edges pointing the SCC source node is replaced with such an exit node and the return edge is removed. Figure 3.7a shows an example in which an SCC (containing the tasks *B, C, D*) is identified and transformed into a BPMN loop sub-process.

The excluded SCCs (those with more than one node with incoming or outgoing edges outside the SCC) are analysed again with the aim of finding sub-SCCs that can be converted into smaller sub-processes. In detail, for each pair of nodes having SCC-external edges, an induced graph is built by considering the node-pair and all reachable nodes



(a) BPMN loop clustering example



(b) Cyclic clustering example

Figure 3.7: Loop clustering examples

having only SCC-internal edges. Such iterative analysis may identify more than one (partially) overlapping sub-SCCs. In this case, the set with the maximum cardinality of nodes is selected. Finally, the SCC components are grouped into a generic cyclic or BPMN loop (if a return edge exists) sub-process.

Figure 3.7b shows an example in which the SCC identified by applying the Tarjan algorithm to the whole graph of the initial process (task set: *B, C, D, E, F, I*) cannot be directly grouped into a sub-process. Indeed, more than one of its tasks has at least one incoming/outgoing sequence flow coming from/leading to a task out of the SCC (e.g., *F*). However, by analysing the SCC subsets, the maximal sub-SCC that is free from any node with SCC-external edges (except for source and target nodes) is identified (task set: *C, D, I*). Since no sequence flow exists between tasks *I* and *C*, the resulting sub-process is not a BPMN loop sub-process.

Sequence Clustering

Sequence clustering identifies all the sequences of tasks in the process that can be grouped into subprocesses. To this purpose the process graph is analysed by applying the generic flow analysis framework with the following setting:

1. $V = 2^N$ where N is the set of graph nodes;
2. the transfer function f is defined as:

$$f(x) \quad = \quad GEN[n] \cup (x - KILL[n])$$

where $GEN[n]$ and $KILL[n]$ are defined as:

$$\begin{aligned} GEN[n] &= \{n \in N : |outE(n)| \leq 1 \wedge |inE(n)| \leq 1\} \\ KILL[n] &= \{m \in N : |outE(n)| > 1 \vee |inE(n)| > 1\} \end{aligned}$$

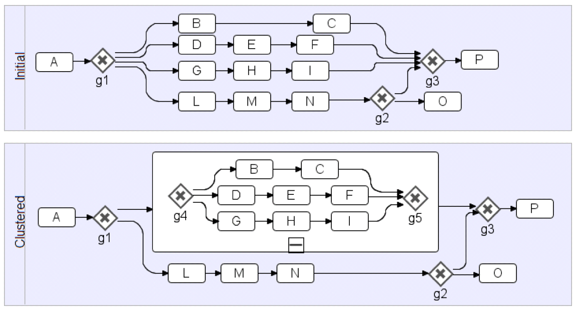


Figure 3.8: Alternative clustering example

and

$$\begin{aligned} outE(n) &= \{(n,m) \in E : m \in N\} \\ inE(n) &= \{(m,n) \in E : m \in N\} \end{aligned}$$

3. $\bigwedge = \cup$, so that $IN[n] = \bigcup_{m \in z(n)} OUT[m]$
4. *forward* $(z(n) = pred(n))$ is the information propagation direction.

This flow analysis allows us to identify all the task sequences. Sequences are then additionally filtered by discarding: (i) each sequence that is not composed of, at least, three tasks; and (ii) in case of overlapping sequences, each non maximal sequence. The output of this clustering technique is a new BPMN process in which the identified task sequences are grouped into sub-processes.

Alternative Clustering

Alternative clustering analyses the process to identify alternative paths (i.e., different sequences of process elements connecting the same pair of decision and merging points) and groups them into sub-processes.

Figure 3.8 shows an example of a process in which an alternative path pattern is identified. The pattern includes three out of the four paths between the two gateways g_1 and g_3 (source and sink of the candidate cluster). The last path (*<L,M,N>*) is not included since it contains elements, e.g., the gateway g_2, on a path that does not pass through the sink gateway g_3 (i.e., this path is not in the set of paths between g_1 and g_3).

Given a process graph, the flow analysis framework is applied two times to the whole graph with different purposes and settings to find the candidate alternative clusters (a cluster is considered a candidate alternative cluster if it is composed of, at least, two alternative paths).

The flow analysis is executed in the graph to compute (i) dominance (a node n_B is dominated by a node n_A if every path from the initial node of the graph, i.e., the root of the graph, to n_B contains n_A) and (ii) postdominance (a node n_A is postdominated by a node n_B if all paths from the node n_A to the graph's leaves, i.e., nodes with no outgoing edges, contain n_B), for all nodes representing BPMN gateways.

The setting required for instantiating the generic flow analysis framework for the dominance analysis is the following:
1. $V = 2^{Gw}$ where Gw is the set of nodes representing gateways;
2. the transfer function f is defined as:

$$f(x) \quad = \quad GEN[n] \cup (x - KILL[n])$$

where $GEN[n]$ and $KILL[n]$ are defined as:

$$\begin{aligned} GEN[n] &= \{n \in N : n \in Gw\} \\ KILL[n] &= \emptyset \end{aligned}$$

3. $\bigwedge = \cap$, so that $IN[n] = \bigcup_{m \in z(n)} OUT[m]$
4. *forward* $(z(n) = pred(n)))$ is the information propagation direction.

The output of the dominance analysis is, for each task-node, the set of gateways that have to be necessarily traversed for reaching the current node from the graph root.

The setting applied for instantiating the generic flow analysis framework to find the postdominators of each node n is the following:
1. $V = 2^{Gw}$ where Gw is the set of nodes representing gateways;
2. the transfer function f is defined as:

$$f(x) \quad = \quad GEN[n] \cup (x - KILL[n])$$

where $GEN[n]$ and $KILL[n]$ are defined as:

$$\begin{aligned} GEN[n] &= \{n \in N : n \in Gw\} \\ KILL[n] &= \emptyset \end{aligned}$$

3. $\bigwedge = \cap$, so that $IN[n] = \bigcap_{m \in z(n)} OUT[m]$
4. *backward* $(z(n) = succ(n))$ is the information propagation direction.

The output of the postdominance analysis is the set of gateways that have to be necessarily traversed for reaching the final nodes from each task-node.

Once dominance and postdominance are known for every node n, they are used for each pair of gateways (g_A, g_B), to find out the $CandidateNodes$ set containing all nodes: (i) dominated by g_A and (ii) postdominated by g_B. For instance, in Figure 3.8 all the nodes contained in the four paths from the source to the sink gateway (respectively g_1 and g_3) are dominated by g_1. However, since the nodes in the path <*L, M, N,* g_2> are not postdominated by the sink gateway g_3 only the nodes in the other paths belong to the $CandidateNodes$ set.

The $CandidateNodes$ set defines a sub-graph of the original process graph, whose edges are all and only the original edges connecting two nodes in the set. When considering this sub-graph, some nodes may become unreachable from g_A or unable to reach g_B. Those nodes are excluded from the alternative pattern through a simple reachability analysis.

Hence, for each pair of gateways, the set of nodes in an alternative pattern between them is identified and their corresponding BPMN elements are grouped into a sub-process. In case of overlapping patterns, the instance with the highest number of tasks is selected.

Page Clustering

Page clustering groups sets of tasks contained in the same Web page into a sub-process by starting from the assumption that elements (e.g., forms) in the same page are related to the same functionality. The initial BPMN process consists of tasks representing the GUI forms and links found in the visited Web pages. Hence, the purpose of this clustering is to group tasks that represent elements defined in the same Web page. Besides the specific tasks, also gateways connected only to tasks in the cluster are added to the page cluster. On the other hand, in case of tasks representing elements shared among more than one Web page (e.g., forms with the same target action and the same field set), no page clustering is applied.

Dependency Clustering

The objective of this clustering technique is optimizing the partitioning level of the process elements, so that the resulting organization simultaneously minimizes *coupling* (i.e., the connections among elements of distinct clusters) while maximizing *cohesion* (i.e., the connections among elements of the same cluster).

Coupling and cohesion are often evaluated in system maintenance and evolution for improving system architectures and source code quality. In our case, their use in process modularization relies on the assumption that strictly connected process elements (i.e., elements characterized by high cohesion among them and low coupling with other process elements) represent logically related activities, so that they can be considered as sub-systems (sub-processes).

Therefore the goal of this type of clustering is, given a graph representing a process, finding a “good” partition of the graph itself. A partition is the decomposition of a set of graph elements (nodes and edges) into mutually disjoint clusters. The partition is a "good partition" when: highly interdependent nodes (elements of the initial process) are grouped in the same cluster and independent nodes are assigned to separate clusters. Once clusters are identified in the graph they can be converted into sub-processes containing process elements corresponding to the graph nodes in the cluster. Additional gateways are added, if required, to the resulting process when converting the clustered graph into the corresponding process.

The following measures [106] for cohesion (A_i) of cluster i and coupling ($E_{i,j}$) between clusters i and j are used to find such a partition in a given graph:

$$A_i = \frac{\mu_i}{N_i^2} \qquad E_{i,j} = \frac{\epsilon_{i,j}}{2N_iN_j}$$

where N_i and N_j are respectively the number of graph nodes in clusters i and j; μ_i is the number of dependencies internal to cluster i; and $\epsilon_{i,j}$ is the number of dependencies between clusters i and j. Since auto-loops on activities cannot occur in processes[2], the denominator of A_i becomes $N_i(N_i - 1)$. A_i and $E_{i,j}$ are between 0 and 1, being 0 when no dependency holds and 1 when there is full connectivity.

The Modularization Quality MQ [106], which will be the objective function of the optimization process, is defined as a measurement of the modularity in terms of process element cohesion and coupling:

$$MQ = \frac{1}{k}\sum_{i=1}^{k} A_i - \frac{1}{\frac{k(k-1)}{2}} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} E_{i,j}$$

where k is the number of graph partitions. If $k = 1$, $MQ = A_1$. The MQ measurement ranges between -1 (no cohesion, maximum coupling) and 1 (no coupling, maximum cohesion). Given a dependency graph, the modularization algorithm partitions the software system so as to maximize MQ. Given a set S that contains n elements, the number $S_{n,k}$ of distinct $k-$partitions (i.e., partitions consisting of k non-empty clusters) satisfies the recursive equation:

$$S_{n,k} = \begin{cases} 1 & \text{if k=1 or k=n} \\ S_{n-1,k-1} + kS_{n-1,k} & \text{otherwise} \end{cases}$$

The $S_{n,k}$ entries grow exponentially with the size of S. For instance, a dependency graph with 5 nodes is associated with 52 distinct partitions, while a graph with 15 nodes is associated with 1,382,958,545 distinct partitions. Hence, the exact optimal solution cannot be found for real and non trivial graphs. Heuristic-based techniques are often

[2]We assume activities can have only one input and output sequence flow.

```
** Hill-climbing clustering algorithm: **
S ← Graph elements
P ← GenerateRandomPartition(S)
repeat
        BNP ← BetterNeighboringPartitions(P)
         if BNP !=0
                P ← SelectRandomly(BNP)
        end if
until P does not change
$P_{max}$ ← P
```

Figure 3.9: Hill climbing algorithm aiming at maximizing the modularization quality of a process

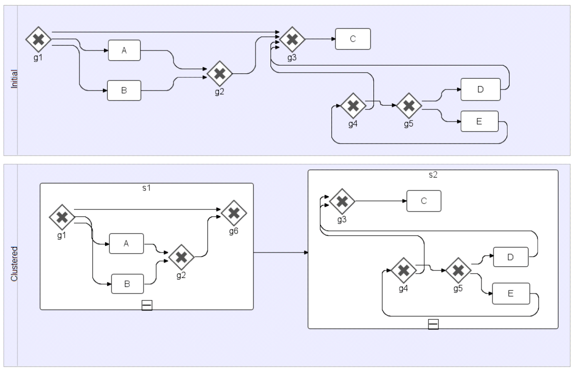


Figure 3.10: Dependency clustering example

applied to identify sub-optimal solutions. Such heuristic techniques use the notion of neighbouring partitions, obtained by moving elements among the clusters of the partition, so as to improve MQ. A partition NP is a neighbour of a partition P if it is the same as P except for a single element that belongs to different clusters in the two partitions.

The clustering process is treated as an optimization problem in which a heuristic-based algorithm tries to maximize the objective function MQ in charge of measuring the "quality" of graph partitions. A pseudo-code for a hill-climbing algorithm is given in Figure 3.9.

In this pseudo-code, a random solution is generated ($GenerateRandomPartition$) and then evolved by considering its neighbouring solutions evaluated by means of the objective function MQ ($BetterNeighboringPartitions$). Evolution is iterated until neighbours exist. Alternative implementations can use genetic algorithms, for example see Mancoridis et al. [106]. When a large graph is analysed, the number of clusters in the (sub-)optimal partition may be large. In this case, it makes sense to group the clusters, thus creating a hierarchy of clusters.

Figure 3.10 shows an example in which the dependency cluster has been applied for grouping process elements. The gateway $g3$ (Figure 3.10 "Initial" pool) is the join point for two high-quality (maximally cohesive and minimally coupled) partitions of the process. The "Clustered" pool in Figure 3.10, shows the clustered process. Notice that, in this process, one gateway ($g6$) has been added in order to make the process consistent. By considering the process elements of the obtained clusters, cohesion (A_{S1}, A_{S2}) and coupling ($E_{S1,S2}$) have been computed as follows: $A_{S1} = \frac{4}{16}$; $A_{S2} = \frac{7}{36}$; $E_{S1,S2} = \frac{2}{2(4)(6)}$; and so $MQ = \frac{A_{S1}+A_{S2}}{2} - \frac{E_{S1,S2}/2}{2} = 0.21$.

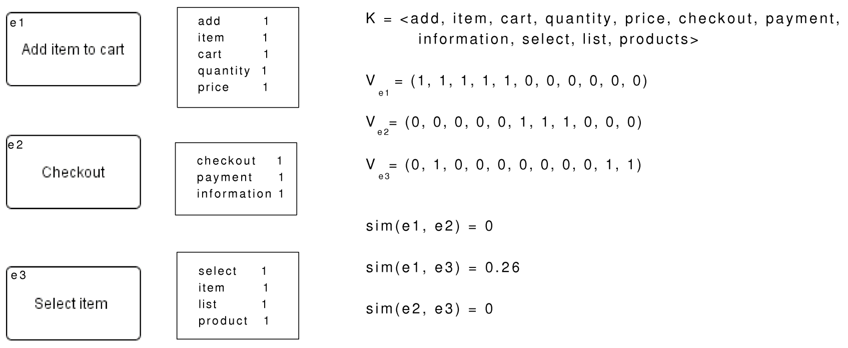

e1
Add item to cart
add 1
item 1
cart 1
quantity 1
price 1
e2
Checkout
checkout 1
payment 1
information 1
e3
Select item
select 1
item 1
list 1
product 1
K = <add, item, cart, quantity, price, checkout, payment, information, select, list, products>
V_e1 = (1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0)
V_e2 = (0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0)
V_e3 = (0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1)
sim(e1, e2) = 0
sim(e1, e3) = 0.26
sim(e2, e3) = 0

Figure 3.11: Term-based analysis example

Semantic Clustering

The objective of this clustering technique is grouping the process elements based on the similarity in their content. The content of a GUI element (form or link) is represented by a list of automatically extracted terms characterizing the text contained in the GUI-element (and/or, in case of forms, in its fields) rendered in a Web page. This clustering technique mainly groups process elements according to their "shared" terms. To this aim a Natural Language Processing technique is applied to determine the terms in the Web page that could be associated to each element and to cluster them.

By means of the analysis of the *content information* traced during the initial process recovery (see Subsection 3.1.3) a list of terms is extracted for each process element e.

Let K be the vector containing all terms extracted from all the application pages and related to the process elements (i.e. the union of terms contained in each of the pages in which a process element is contained), with each term uniquely represented by a single entry. A feature vector V_e is built for each element e, with $V_e[i]$ holding the weight of the term $K[i]$. A measure of this term-weight is based on the presence of the term itself $K[i]$ in the element content. When a term is present in the content element e, the related entry $V_e[i]$ in the feature vector is 1; 0 otherwise. More sophisticated metrics for the term weights (e.g., term frequency and inverse document frequency) could be also used.

For example, let us consider the content of three fictitious elements $e1$, $e2$ and $e3$ of a process. Boxes in Figure 3.11 report terms and occurrences for those elements (e.g., the term "add" occurs one time in $e1$).

Given the description of each element e in terms of its feature vector V_e, it is possible to exploit similarity or distance measures to agglomerate entities into clusters. Similarity/distance between clusters is generalized from the similarity/distance between entities by means of the complete linkage rule (different rules such as average linkage could be also applied). According to this rule, the distance between two clusters is computed as the minimum similarity between pairs of cluster elements (this measure privileges cluster cohesion over coupling). We preferred a similarity measure over a distance measure, because the latter is prone to the known problem of sparse or empty vectors: distances become small not only when vectors are close to each other, but also when they are very sparse (or empty), thus leading to the creation of inappropriate clusters. The similarity measure used with the feature vectors described above is the normalized vector product, given by:

$$sim(e1, e2) = \frac{\langle V_{e1}, V_{e2} \rangle}{\|V_{e1}\| \, \|V_{e2}\|}$$

where V_{e1} and V_{e2} are the feature vectors of elements $e1$ and $e2$ respectively; angular brackets indicate the scalar product, which is normalized by the product of the norms, thus giving a similarity measure which ranges from 0 to 1 (under the hypothesis of non-negative weights). After executing the agglomerative clustering, a proper cut point needs to be manually selected. The possibility for the user to choose a given abstraction level (number of clusters or, equivalently, cut point), and then to adjust it toward the top of the hierarchy (less clusters with more elements inside) or toward the bottom (more clusters containing fewer elements) can be an important interactive facility for improving the clustering operation.

In our example, by applying the semantic clustering to the three elements shown in Figure 3.11 and choosing number of clusters equal to 2, elements $e1$ and $e2$ can be grouped into a unique sub-process since their content similarity is higher than the content similarity of other element combinations. More sophisticated Natural Language Processing approaches could be further investigated for improving this semantic clustering technique. For instance, different term extraction, grouping criteria and language analysis algorithms may be considered.

3.1.5 Cluster Labelling

As suggested by Mendling et al. in their studies [111, 112], labels of process tasks and sub-processes impact the level of process readability and understandability.

In our reverse engineering technique, a term-frequency analysis (based on term frequency-inverse document frequency, TFIDF [107]) is applied to the process element (task and sub-process) labels for identifying "representative" terms for each cluster. The assumptions behind this choice are that: (i) a natural language processing technique can be applied to the labels of process tasks for identifying terms characterizing each sub-process (cluster of tasks), and (ii) these terms are adequate to be labels of such sub-processes. These assumptions will be further investigated and evaluated by means of empirical analyses.

Generally, TFIDF is a statistical measure used to evaluate how relevant a term is in a document contained in a document collection. The relevance of a term increases proportionally to the number of times the term appears in the document (term frequency) but is offset by the frequency of the word in the collection of documents. A term, in fact, can occur frequently in a document, as well as in all the others, thus resulting to be a general term rather than a specific term characterizing that document. In order to avoid this kind of situation, the frequency of a term in a document is compared with the average frequency of that term in the whole set of documents by offsetting its frequency with the inverse document frequency.

Term frequency ($TF_{i,j}$) of a term t_i in the document d_j is defined as follows:

$$TF_{i,j} = \frac{n_{i,j}}{\sum_k k_{k,j}}$$

where: $n_{i,j}$ is the number of occurrences of term t_i in the document d_j and $\sum_k k_{k,j}$ is the sum of the occurrences of all terms in document d_j.

The inverse document frequency (IDF_i) of the term t_i is defined as follows:

$$IDF_i = log\frac{|D|}{d(t_i)}$$

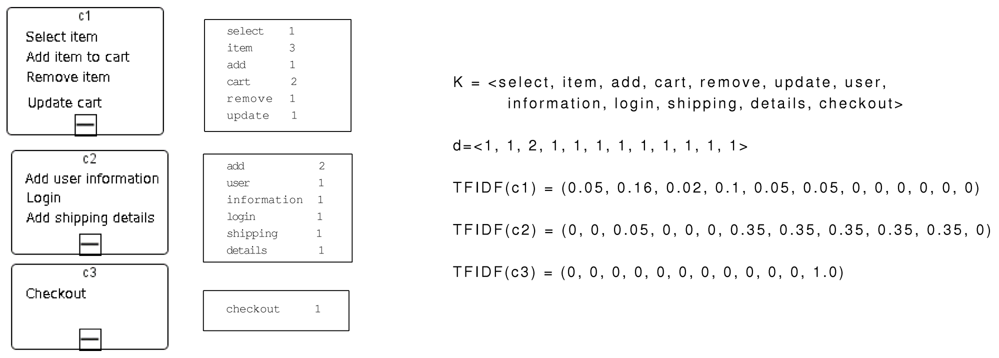
where: $|D|$ is the number of documents in the collection; and $d(t_i)$ is the number of documents in which the term t_i occurs.

Hence, TFIDF for the term t_i in the document d_j is computed as follows:

$$TFIDF_{i,j} = TF_{i,j} * IDF_i$$

In our case, a document is the union of the sets of terms extracted from the labels of the elements (tasks, sub-processes and nested elements) in each process cluster. The complete “dictionary” of terms derived from the document collection is composed of all terms contained in labels of the entire set of process elements.

Figure 3.12 shows an example of TFIDF computed for three process clusters $c1$, $c2$ and $c3$. $TFIDF(c)$ represents the vector of TFIDF values of terms characterizing the cluster c. For example, the weight of the term *add* in the vector $TFIDF(c2) = 0.05$ is lower than all other positive (and non-zero) weights in $TFIDF(c2)$, though it would be the highest weight when calculated using only the term frequency. The term *add*, in fact, is not specific of the element $c2$, since it is shared between $c2$ and $c1$.



```
select   1
item     3
add      1
cart     2
remove   1
update   1

add          2
user         1
information  1
login        1
shipping     1
details      1

checkout   1
```

```
K = <select, item, add, cart, remove, update, user,
       information, login, shipping, details, checkout>

d=<1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1>

TFIDF(c1) = (0.05, 0.16, 0.02, 0.1, 0.05, 0.05, 0, 0, 0, 0, 0, 0)

TFIDF(c2) = (0, 0, 0.05, 0, 0, 0, 0.35, 0.35, 0.35, 0.35, 0.35, 0)

TFIDF(c3) = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.0)
```

Figure 3.12: Example of TFIDF values

By exploiting the TFIDF value computed for each term of the labels associated to the clusters, we identify the terms characterizing the whole clustered sub-process as those having highest TFIDF. In detail, the following steps are used for finding representative sub-process terms:

1. Identification of labels of process elements (tasks, sub-processes and nested elements) contained in a cluster c;
2. Analysis of each identified label and decomposition into terms. For example, the label “Add item to cart” is decomposed into three main terms: “add”, “item” and “cart”. In this step, stop words (e.g., “to”, “the”) are removed;
3. Creation of a common dictionary of terms as the union of the identified terms;
4. Creation of the vector $TFIDF(c)$ for each cluster c of the process, by computing the TFIDF value for each term in the dictionary;
5. For each cluster c, ranking of its terms according to their TFIDF values and selection of those with the highest TFIDF value (e.g., the top 3). The terms are provided to the analyst as suggestions for the creation of a meaningful sub-process label.

3.1.6 The tool

We developed *JBPRecovery*, a tool supporting the proposed reverse engineering approach and implementing all the introduced clustering methods. It is composed of the following six logical modules:

- **Tracer**: a Javascript extension of the Mozilla Firefox browser[3] tracing the execution of

[3]Mozilla Firefox provides special features that can be used by its extensions `https://developer.mozilla.org/En` with the aim of simplifying the implementation of browser utilities.

a target Web system.
- **Process Constructor**: a Java module analysing the execution traces with the aim of extracting a BPMN-based description of the process implemented by the target application.
- **Cluster Detector**: a Java module implementing the clustering techniques. It provides sub-modules realizing respectively structural, page-based, dependency-based and semantic clustering. Configuration files are used to select clustering types and their application order. This module takes as input a graph representing the process under analysis, it applies the selected clustering technique/s and it produces a list of clusters, each grouping a set of graph nodes (process elements).
- **Label Generator**: a Java module implementing the labelling suggestion approach. It applies TFIDF to sets of terms in order to find those regarded as representative for each clustered sub-process. This module is used by the Cluster Detector module to suggest terms for labelling clustered sub-processes.
- **Format Converter**: a Java module converting BPMN-based processes into their related intermediate graph representation (and vice versa); it is mainly used by the Cluster Detector module.
- **Utility Provider**: a Java module providing utilities to other modules. Examples of provided functionalities are setting and configuration management, reading/writing from/to files and the user interface.

Moreover, the following existing tools are used by the *JBPRecovery*'s modules:
- **Bunch**[4]: a clustering tool for software systems. Bunch allows to evaluate the quality of application modularization, by analysing a source code graph modelling code dependencies. It is used by the Cluster Detector module for realizing the dependency-based clustering.
- **RapidMiner**[5]: a tool for machine learning and data mining. It provides several operators (e.g., for agglomerative/hierarchical clustering, Support Vector Machines and Meta Learning) for data mining, data visualization and an IDE to define operator trees (a tree is a composition of operators) for data analysis. RapidMiner is used by the Cluster Detector module for realizing the semantic clustering by means of a specialized composition of operators (implementing the semantic clustering technique introduced in Subsection 3.1.4).
- **WVtool**[6]: Java library providing special capabilities for statistical Natural Language Processing. It is used by the Label Generator module for computing the TFIDF value of

[4]`http://serg.cs.drexel.edu/redmine`
[5]`http://sourceforge.net/projects/yale`
[6]`http://nemoz.org/joomla/content/view/43/83/lang,de/`

terms and suggesting representative clustered sub-process terms.

3.1.7 Reverse Engineering Technique Evaluation

A case study has been conducted, by applying the proposed approach to real Web applications. The goal of the case study is investigating the applicability of the proposed reverse engineering techniques in recovering business processes exposed as Web applications and modularizing them.

Three e-commerce applications (either downloadable or available on-line) have been selected as subjects of our case study: Softslate[7], Erol[8] and Communicart[9]. The applications implement shopping carts that allow the user to manage on-line stores. They realize functionalities to support the on-line retail of products (e.g., catalog, cart, order form and payment checkout management), systems for handling customer accounts (e.g., user account login and registration) and product shipping. All the applications represent medium Web systems — in terms of application size and complexity — developed by adopting different languages and technologies (mainly Java/Jsp and PHP), using a database to store information about the product catalog, carts and users. For all the three applications, fully-functioning demo versions can be accessed on-line on the respective application Web sites.

Research Questions and Metrics

The main aim of the case study is to evaluate in real applications the effectiveness of the proposed reverse engineering and clustering techniques in respectively recovering processes and modularizing them for improving process readability. To this aim we tried to answer the following research questions:

RQ1 *What is the accuracy of the reverse engineering technique, in terms of over-approximation and under-approximation?*

RQ2 *What is the quality of the modularization produced by each clustering technique?*

RQ3 *What is the usefulness of the labelling suggestion technique, in suggesting meaningful terms for clustered sub-processes?*

RQ1 deals with the ability of the reverse engineering technique in recovering "reasonable" processes for the analysed application. To evaluate this ability we estimated over and under approximation of the recovered processes. There is under-approximation

[7]`http://www.softslate.com`
[8]`http://www.eroldemostore.co.uk`
[9]`http://www.communicart.biz`

when relevant process tasks (*missed tasks*) are not captured/modelled in the recovered processes. There is over-approximation when non-business process tasks (*misidentified tasks*) are part of the recovered processes. To measure under and over approximation, an expert manually analysed each application (i.e., he ran the application and analysed its documentation and code, if any) with the aim of identifying the implemented process tasks. According to works in the literature [82], we used the result of this manual analysis as a "gold standard" and we compared it with the recovered process by computing the following accuracy measures:

$$Recall = \frac{|identified\ tasks \bigcap gold\ standard\ tasks|}{|gold\ standard\ tasks|}$$
$$Precision = \frac{|identified\ tasks \bigcap gold\ standard\ tasks|}{|identified\ tasks|}$$
$$F - Measure = \frac{2*Precision*Recall}{Precision+Recall}$$

Recall gives an idea about the ability of the technique in detecting all business tasks in the application. A high recall corresponds to a limited number of missed business tasks. $Precision$ gives an idea about the ability of the technique in discarding non-business tasks (misidentified tasks). A high precision corresponds to a limited number of non-business tasks in the recovered process. Finally, the $F - Measure$ provides an overall evaluation by evenly weighting recall and precision.

RQ2 deals with the ability of clustering techniques in modularizing recovered processes according to "good modularization" principles. For the evaluation of this ability, we manually analysed each sub-process identified by individually applying each clustering technique and we classified it in terms of "well-clustered" or "not-well-clustered". Hence, by determining the fraction of "well-clustered" sub-processes the following clustering quality measure has been computed:

$$ClustQuality = \frac{\#well\text{-}clustered\ sub\text{-}processes}{\#evaluated\ sub\text{-}processes}$$

ClustQuality gives an idea about the ability of a specific clustering technique in grouping process elements. Higher clustering quality corresponds to a limited number of sub-processes not adequately clustered. When manually deciding if a sub-process reported by a clustering technique is to be considered "well-clustered" or "not well-clustered", we based our decision on the answers given to the two following questions: How well does the recovered sub-process model a subsystem? How well does the recovered sub-process model the functionality/behaviour provided by the subsystem?

RQ3 deals with the ability of the labelling suggestion technique in selecting meaningful terms for clustered sub-processes. For this evaluation we estimated the "appropriateness" of the suggested terms. To this aim, we manually classified each set of suggestions automatically provided by our technique (a set of suggestions for a sub-process contains the top 3 terms in the TFIDF ranking) as "meaningful" or "non-meaningful", where "meaningful" means that at least one of the terms in the list is expected to be used in constructing a full label for the sub-process. In this case, the automatic suggestion is considered "meaningful". The following suggestion quality measure has been computed:

$$MeaningfulSuggestion = \frac{\#meaningful\ suggestions}{\#evaluated\ subprocesses}$$

MeaningfulSuggestion gives an idea about the ability of the technique in selecting "appropriate" terms. A higher ratio of meaningful suggestions corresponds to a reduced effort for the analyst in identifying meaningful sub-process labels able to improve the process readability.

Execution

In the case study, the following steps have been performed for each considered application:

1 *JBPRecovery* has been used to trace some executions of the application with the aim of exercising each application functionality, at least, once;

2 The traces have been analysed for building the initial BPMN business process;

3 *JBPRecovery* has been used to refine the initial process by introducing sub-processes for grouping related tasks. The clustering techniques have been applied in different orders and combinations with the purpose of evaluating their effectiveness in improving the process structure;

4 The application has been manually analysed for collecting the "gold standard" (in terms of tasks, clustered sub-processes and labels);

5 The required measures (e.g., number of "well-clustered" sub-processes and "meaningful" suggested terms) have been computed for each process extracted, as described above.

Results and Discussion

JBPRecovery has been used to trace, on average 8 executions (10, 7 and 7 respectively for Softslate, Erol and Communicart) per application with the aim of covering/exercising

App.	**Tasks** (client)	**Gat.s**	**Events**	**Seq. Flows**	**Msg. Flows**	**Rec.** %	**Prec.** %	**F-Measure** %
Softslate	100 (87)	24	31	155	27	96	87	91
Erol	59 (31)	23	34	166	30	82	74	77
Communicart	61 (41)	28	32	160	28	81	85	82
					average (dev.std)	86 (8)	82.5 (7)	83 (7)

Table 3.1: Initially recovered processes and their (task) under/over approximation

each implemented functionality at least once. On average, 20.5 user actions (13.7, 15.3 and 32 respectively for Softslate, Erol and Communicart), i.e., link and submit clicks, have been performed during each execution. Table 3.1 (left) summarizes the number of process elements (e.g., tasks, gateways) composing the initially recovered processes. On average, each process contains 314 process elements and 33% of them are tasks.

***RQ1**: Process under and over approximation*

Table 3.1 (right) presents the results obtained for *Recall*, *Precision* and $F-Measure$ computed for the recovered processes. We can observe that the $F-Measure$ is high (on average 83%). *Recall* and *Precision* are high too and they show that under and over approximation of the recovered processes are quite limited.

The obtained *Recall* is, on average, 86% and it is always higher than 81% (the lowest value has been obtained for Communicart). This suggests that business tasks realized by a Web application can be adequately inferred by tracing the execution of its GUI elements. Furthermore, the obtained *Precision* is, on average, 82.5% and it is always higher than 74% (the lowest value has been obtained for Erol). This suggests that discarding application non-business tasks by considering sets of GUI elements is the most challenging activity. In other terms, links and forms seem to be effective in identifying all business tasks even though (slightly) less effective in discarding non-business tasks.

***RQ2**: Process modularization*

Table 3.2 (left) presents the results obtained by clustering the initially recovered processes. The column "Clustering" of the table shows combinations and orders of the clustering techniques considered in the experiment. We applied structural (loop clustering followed by the alternative one), dependency and semantic clustering individually, and the three-combinations (without repetitions) of these clustering techniques in all possible different orders. We do not report the results of the page clustering technique since the limited number of pages of the considered Web applications does not allow a flexible combination of this technique with the others. Columns "Clustered Subprocesses" and "ClustQuality" of the table summarize the number of the identified clusters and the quality measure of such clusters according to the expert's evaluation. We can observe that, on average, we identified 8.6 clusters of tasks per process (10, 7.4 and 8.4 respectively for Softslate, Erol and Communicart) and that, by combining the three clustering techniques, the number of identified clusters is double, on average, than individually applying each of them. Moreover, the measured cluster quality evaluated according to the expert's opinion, is on average 69.7% and it seems to be reasonably adequate for automatically recovered and clustered processes. Table 3.2 (left) shows that the dependency clustering obtained the best performance (in terms of higher number of identified clusters and cluster quality evaluation) with respect to the other two clustering techniques individually applied. Among the considered clustering technique combinations and orders, the structural-dependency-semantic and structural-semantic-dependency clusterings obtained the best performance.

***RQ3**: Clustering labelling*

Table 3.2 (right) presents the results obtained in the evaluation of the terms automatically suggested for the clustered sub-processes. The expert evaluated the suggestions proposed for 283 clustered sub-processes and, on average, for 70.3% of them, the suggestion has been classified as "meaningful". The obtained result indicates that the use of a term-frequency analysis applied to task labels is an adequate starting point for identifying sub-process labels that improve the process readability.

Overall Considerations

According to the overall results of the experiment we can notice that a limited number of application executions (10, 7 and 7 for Softslate, Erol and Communicart, respectively) are adequate to reach a reasonable accuracy of the recovered processes (limited under/over approximation). Hence, a stable process (in terms of number of its components) with a high accuracy can be obtained in a short time. This is particularly interesting when a fast comprehension of the application is required (e.g., for maintenance activities).

In the experiment we notice that GUI links and forms seem to be effective in identifying all application business tasks even though slightly less effective in discarding application non-business tasks. This result gives us confidence that the recovered processes actually represent the processes underlying Web applications. Furthermore, the structure of the application seems to affect the recovered process accuracy. The best accuracy (highest

App.	**Clustering Combination**	**Clustered Subprocess #**	**ClustQuality %**	**MeaningfulSuggestion %**
Softslate	St	6	100	100
	Dep	7	71	85
	Sem	6	83	100
	St-Dep-Sem	14	85	71
	St-Sem-Dep	15	86	73
	Dep-St-Sem	8	87	87
	Dep-Sem-St	11	63	81
	Sem-St-Dep	12	83	91
	Sem-Dep-St	12	66	75
	average (dev.std)	10 (3)	80.4 (11)	84.7 (11)
Erol	St	3	66	100
	Dep	7	57	57
	Sem	4	75	50
	St-Dep-Sem	11	63	81
	St-Sem-Dep	10	70	60
	Dep-St-Sem	7	66	66
	Dep-Sem-St	9	44	55
	Sem-St-Dep	9	66	44
	Sem-Dep-St	7	66	57
	average (dev.std)	7.4 (3)	63.6 (8)	63.3 (17)
Communicart	St	5	80	80
	Dep	8	65	75
	Sem	2	50	100
	St-Dep-Sem	11	81	81
	St-Sem-Dep	10	71	71
	Dep-St-Sem	10	50	66
	Dep-Sem-St	8	62	50
	Sem-St-Dep	10	50	75
	Sem-Dep-St	12	57	57
	average (dev.std)	8.4 (3)	62.8 (12)	73.7 (15)
	average (dev.std)	8.6 (3)	69.7 (13)	70.3 (16)

Table 3.2: Quality of clustered sub-processes and meaningfulness of suggested labels
St = Structural (Loop-Alternative), Dep = Dependency and Sem = Semantic clustering

$F - Measure$) has been obtained for Softslate, whose Web-GUI uses more forms than links for submitting data and requests to server components. Communicart uses a mix of forms and links (links and forms are evenly distributed), while Erol, which had the lowest accuracy, uses mainly links. We can therefore hypothesize that forms positively influence the accuracy of the process.

The experiment results show also that process clustering is effective in reducing the overall size and complexity of the recovered processes by grouping process tasks. In particular, the combination of the three clustering techniques increases the process modularization. By inspecting the clustered processes we found that a possible reason for this difference in process modularization is that the overlapping of process tasks, clustered by applying each clustering technique (structural, dependency and semantic based) individually, is quite limited. Moreover, the cluster quality evaluation reveals that the quality of the identified clusters is reasonably high. In the experiment, the ClustQuality is higher than 50% (60%) for 85% (77%) of the 27 differently clustered processes. As in the case of the process accuracy evaluation, the structure of the application (number of forms with respect to links) seems to influence the quality of the modularization obtained by applying structural and dependency clustering techniques. For example, by applying structural clustering only, Softslate obtained the highest value for the ClustQuality measure while Erol scored the lowest. This result is reasonable considering that GUI-forms are often used to guide the user application navigation in realizing specific execution flows (e.g., providing services), thus “structuring” the application underlying process.

According to both quantitative (i.e., based on the number of clustered sub-processes) and qualitative (i.e., considering how the process elements have been clustered) analysis, the structural-dependency-semantic and structural-semantic-dependency techniques seem to be the more adequate for increasing the understandability of recovered processes. On one side, they have the higher percentage of identified process clusters (on average, 75% more than processes clustered with a single clustering technique). On the other side, according to the expert’s opinion, they better capture relevant application macro-functionalities (e.g., user login, item selection, cart management and checkout) by describing them as sub-processes. Therefore, the overall process obtained by applying such clustering techniques results to be more readable and understandable. Figure 3.13 shows the client-side part of one of them: the Softslate process clustered by applying structural-semantic-dependency technique. Due to space limitation, in the figure some of the clustered sub-processes are shown by means of collapsed BPMN sub-processes, while two of them (representing user login/registration and cart management) are expanded showing their internal structure, including also nested clusters.

Finally, the experiment results seem to confirm that meaningful labels can be identified for the clustered sub-processes by analysing the labels of the tasks in those sub-processes. We can notice that some statistical correlation exists in the experiment results between the number of meaningful suggestions and the cluster quality (last two columns in Table 3.2). By applying the *Pearson's correlation*[10] [88] between ClustQuality and Meaningful-Suggestion we obtained $P > 0.4$ and $p - value = 0.03$. Hence, the obtained correlation indicates that the meaningfulness of the suggested labels improves when increasing the quality of the clustered sub-processes.

Summarizing, the experiment confirms the intuitions that: (a) Web applications forms and links implicitly contain information about the underlying application process; (b) dynamic analysis of the Web-GUI can be used to infer this underlying process; and (c) clustering techniques can be successfully applied to the recovered process to abstract and better modularize it.

Threats to Validity

The reduced number of subjects of the case study and their unique application domain (e-commerce) is one of the threats to validity that limit the generalization of the obtained results (*external validity*). However, we found the considered applications quite representative in the Web for such a domain. The e-commerce domain is one of the most well-known domain for Web applications that expose business processes. Different application domains will be considered in the next experiments.

Another threat to the external validity is related to the application executions traced to recover the initial BPMN-based processes. Different processes can be obtained by considering different sets of traces. We tried to limit the impact of this factor by applying a functionality-based coverage criterion for selecting application executions. Further investigations will be devoted to verify the effect of different sets of traces in the obtained processes.

Additional threats involve measures used to answer the research questions (*internal validity* threats). For **RQ1**, we only considered over and under approximation of the recovered processes in terms of "task coverage" [82] with respect to the gold standard. This coverage criterion is used to compare models when exhaustive criteria cannot be

[10]The *Pearson's correlation coefficient* reflects the degree of linear relationship between two variables and ranges from +1 to -1. A correlation of +1 means that there is a perfect positive linear relationship between variables, while a coefficient of 0 means no correlation.

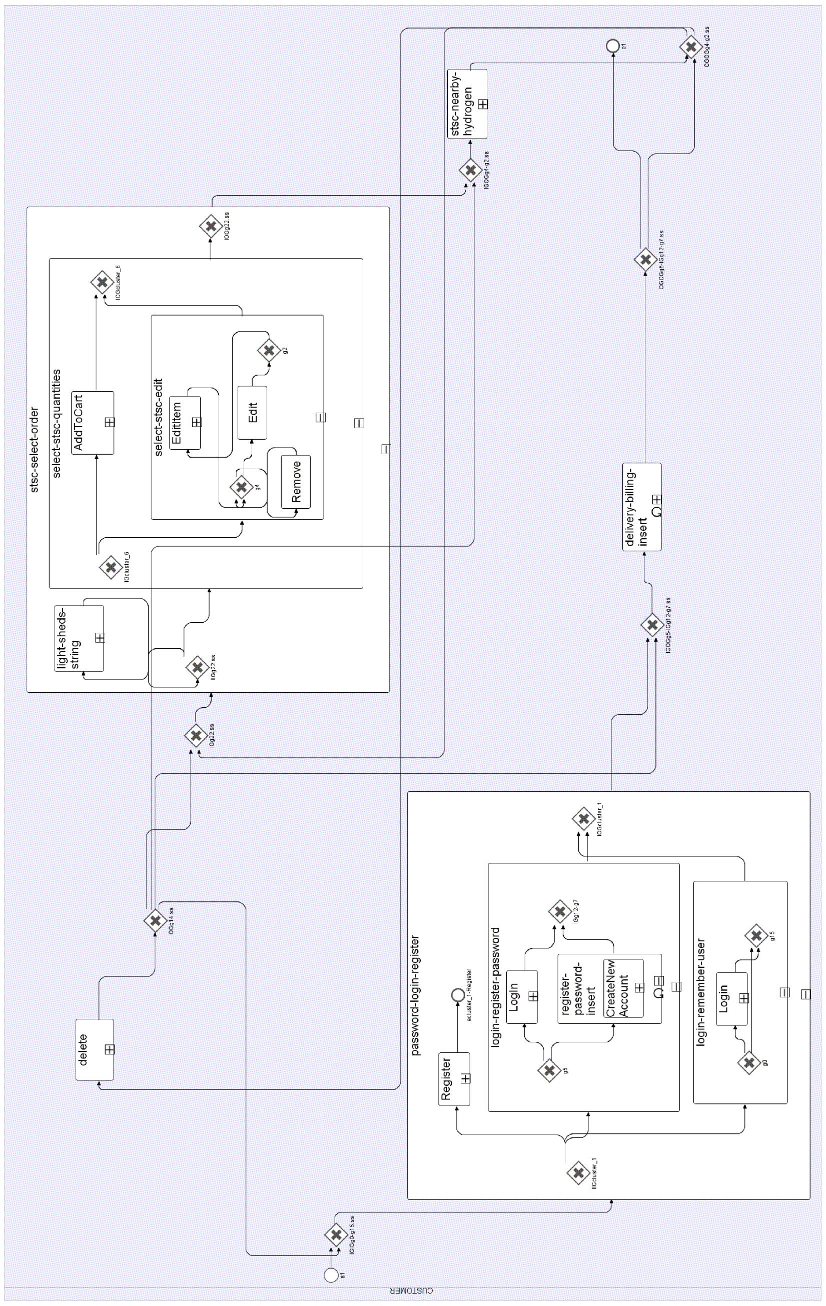


Figure 3.13: The recovered Softslate process clustered by applying structural-semantic-dependency clustering.

applied [82]. Furthermore, strong subjectivity characterizes answers to the research questions. For instance, in answering **RQ2**, an expert is asked to analyse each sub-process. In this task, expert skills and expertise influence the final result. In further evaluation we plan to involve more than one expert with the aim of limiting the subjectivity influence.

Even though these threats to validity limit the obtained results, the overall case study outcome is encouraging, since it indicates a strong potential of the presented techniques in recovering models useful for better documenting and understanding business processes implemented by Web systems.

3.2 Understandability Metrics

Among purposes and uses of business process models, their role in activities directly involving humans (e.g., they facilitate the communication between designers and implementers, they represent a form of documentation of the processes enacted by companies) is of primary importance, thus demanding for process models readable and understandable. Nevertheless, as shown in the previous section, when process models are recovered from executions, they can be very large, complex and intricate (i.e., "spaghetti" processes [185]), thus resulting difficult to read and understand. Model understandability is hence a key factor for maximizing the quality of recovered processes, and hence their effectiveness and usefulness.

Although a lot of effort has been spent in proposing new process recovery techniques, to the best of our knowledge, not so much work has been devoted to study and validate an overall and shared means to evaluate the understandability of the recovered processes. In fact, existing works in the literature focus on: the identification of factors making process models understandable (e.g., personal factors and model characteristics [111]); the analysis of specific factors impacting the understandability (e.g., control-flow complexity [28] and process modularization [149]); the evaluation of the understandability in generic process models (e.g., [110]). However, an overall set of empirically validated process metrics that can be easily and automatically applied to early evaluate the understandability of recovered business process models lacks.

In this chapter, we analyse and empirically evaluate a set of surveyed metrics for the assessment of process understandability. In detail, we collect and customize a set of process metrics related to the process understandability (Subsection 3.2.1) and we conduct and experimental study to evaluate their effectiveness as early indicators of the process understandability (Subsection 3.2.2).

3.2.1 Process Metrics

In the rest of this subsection, we present a three-layer view for evaluating process model understandability according to relevant model characteristics. In detail, such a view is built by applying a bottom-up approach in which we: (1) survey existing metrics for process models; (2) group the metrics according to the process properties and artefacts they evaluate; and (3) hierarchically organize the groups of metrics. In the following we present the most relevant factors (layer one) and properties (layer two) concerning the characteristics of recovered process models that impact the understandability of this type of models. Moreover, we present a set of metrics (layer three) collected by inspecting existing literature and customized to be used with BPMN process models (some of the metrics were originally proposed for different process languages such as EPC and Petri Nets).

Process Understandability Factors

According to the literature, the most relevant high-level factors impacting the understandability of recovered process models are related to:

- **Term**: the quality of process element labels [113]. To evaluate the quality of a process element label we need to answer the following questions: Is the label composed of relevant and clear terms? Does it have a real meaning for human modellers?
- **Structure**: the quality of the process structure. To evaluate the quality of the process model structure we need to answer the following questions: Is the process structure clear and well organized? Is the overall process model reasonably simple and readable for a human? Although several languages, syntactical process elements and representations can be used to model a process, it has been shown (e.g., [28] and [149]) that some model characteristics, as for example the process model modularity or the number of alternative flows, can influence model quality and effectiveness.
- **Conformance**: the adherence of the recovered models to the actual processes according to the information used to generate them. Studies in the literature (e.g., [173]), in fact, indicate that the model conformance is strongly related to the completeness of the recovered process models and its ability of describing and representing the actual process. Different recovery algorithms and algorithm settings can influence the conformance of the recovered models with the initial artefacts. In

the evaluation of the process conformance, relevant questions can be: How does the process model capture the information used to generate it? How much does the process model generalize the information used to generate it?

Process Understandability Properties

For each of the factors introduced above, a set of properties can be identified. In detail:

- **Term**. Activity labels are a way to provide process models with domain knowledge, thus improving the process model understandability. Previous studies (e.g., [113]) proved that quality aspects in process activity labels are mainly related to label **size** (in terms of word number) and **ambiguity**. In fact, an activity label can be composed of a limited number of terms (i.e., key concepts coming from the domain of the modelled process) or can be a complete-sense sentence. Furthermore, labels can be composed according to different styles (e.g., verb-object versus action-noun) and can contain ambiguous terms (e.g., terms subjected to the *0-derivation property*, i.e., the same term can be used either as verb or noun), thus making its comprehension more difficult. According to the results reported by Mendling et al. [113], both a too high/low label size and a high label ambiguity can potentially compromise the understandability of the process model.

- **Structure**. The most relevant issues with respect to the understandability of process models are recognized to be related to the process structure, organization and design. Hence, the process **size** (e.g., the number of elements in the process), the complexity of the process **flow** (e.g., the complexity of the process control-flow), and the overall **structure** (e.g., the structuredness of the process flow) of the process model can impact its understandability, readability and effectiveness in representing the process. Therefore, an increase in the process size or in the flow and structure complexity can potentially compromise the understandability of the process model; on the other hand, an increase of the flow structuredness can improve the understandability.

- **Conformance**. A process model recovered from existing artefacts needs to “fit” with them, i.e., the model must conform to the artefacts. In other terms, the model needs to be able to represent the whole (structural and behavioural) information contained in the artefacts. However, it is well-known that a recovered model can introduce some degree of generalization/approximation in the model with respect to the initial artefacts, e.g., by adding or removing behaviours and structural information. The introduced generalization/approximation, which can be due for example to spurious information, can potentially compromise the understandability of the process model, and hence has to be limited.

Process Understandability Metrics

In order to measure the process understandability factors and properties, we present here a set of metrics, surveyed from works existing in the literature and customized for our research (e.g., with respect to the specific process language we use, BPMN). Each metric is presented by using the following pattern: metric name, references to introductory works, informal metric presentation, formal metric description, and its expected relation with the process understandability properties (i.e., the hypothesis to be validated) of recovered process models. In the rest of this subsection we introduce the suite of metrics. The three event-based traces $t1$, $t2$ and $t3$ in (3.1) below (each t is a trace, i.e., a list of exercised GUI-elements) and the process models $PM_{example1}$ and $PM_{example2}$ shown in Figure 3.14 are used hereafter as running examples. The $PM_{example1}$ and $PM_{example2}$ are the same models used in Figure 3.7a of Subsection 3.1.4 for illustrating the loop modularization; in this case, however, actual names are provided for GUI-events and, hence, for process activity labels, based on a simple process for order management. By assuming that each page contains only one GUI-element and hence that it is not necessary tracing also the page information (e.g., A represents the pair $\langle A\ page, A\rangle$), the process reported on top of Figure 3.14 is inferred by applying the reverse engineering technique presented in Section 3.1 to traces $t1$, $t2$ and $t3$. The process at the bottom of Figure 3.14 represents the same process, modularized according to the loop pattern.

$$\begin{aligned} t1 &:< A, B, C, D, E > \\ t2 &:< A, B, C, D, B > \\ t3 &:< A, B, C, D, B, C, D, E > \end{aligned} \tag{3.1}$$

- ***Label length*** (Ll). It refers to the average number of words used per activity label in a process P.

$$Ll(P) = \frac{\sum_{l \in Labels} |words(l)|}{|Labels|}$$

where $words(l)$ is the set of words of the label l. $Ll(P) \geq 1$; by assuming that each activity contains at least a word, $Ll(P)$ is equal to 1 if each label contains only one word,

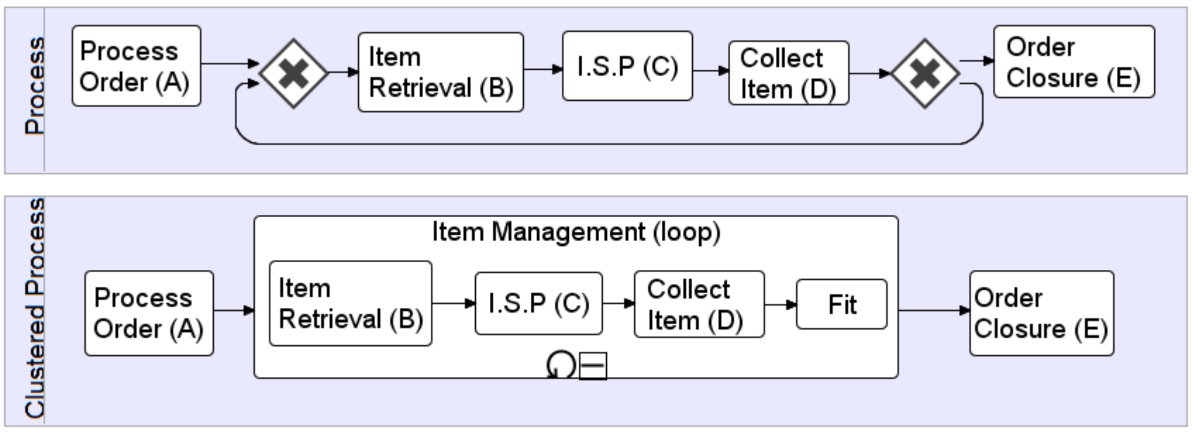


Figure 3.14: $PM_{example1}$(top) and $PM_{example2}$(bottom) process models

otherwise it is > 1. We expect that a too low or high value of the metric indicates a too short or long label. For example, the Ll of $PM_{example1}$ is equal to 1.8 , since 4 activity labels have length equal to 2 and one equal to 1.
- ***Label 0-derivation*** ($L0d$). It refers to the number of words in the labels suffering the 0-derivation property (i.e., words that can be either a noun or a verb depending on the context).

$$L0d(P) = \frac{\sum_{l \in Labels} has\_a\_0\_derivation\_word(l)}{|Labels|}$$

where $has\_a\_0\_derivation\_word(l)$ has value 0 if no word in the label l is a 0-derivation word, 1 if there exists at least a 0-derivation word in $words(l)$. $0 \leq L0d(P) \leq 1$; $L0d(P)$ is 0 if no labels contain 0-derivation words, while it is 1 if all the labels contain at least one 0-derivation word. We identify words suffering the 0-derivation property by automatically consulting the Wordnet dictionary [118] and by requiring the human decision in case of words not included in the dictionary. We expect that a high value of the metric indicates a high ratio of ambiguous labels. For instance, in case of $PM_{example1}$, $L0d(PM_{example1}) = 0.4$ since two labels “Process Order” and “Order Closure” contain 0-derivation words: both “Process” and “Order” in the first case (thus leading to a possible ambiguity in the semantics of the short sentence, that could be interpreted either as “process an order” or “order a process”), only “Order” in the second.
- ***Label style*** (Ls). It refers to the style of the terms used in activity labels. Since activity labels are usually defined in the verb-object or in the action-noun form [116], their style can be classified according to one of the following categories: (vo) verb-object, (an) action-noun, and (rest) other. Ls measures the weighted distribution of the process

labels in these three categories. Weights are assigned to words according to the assumed ambiguity of their category, i.e., a weight of 3 is assigned to terms in the category (rest), 2 to those in the category (an) and 1 to words in the (vo) group. It has been shown [116], in fact, that the (vo) form in process labels is the less ambiguous and more useful form, followed by the (an) form and by the (rest) style.

$$Ls(P) = \frac{3*|rest|+2*|an|+1*|vo|}{3*|Labels|}$$

where vo, an and $rest$ are respectively the set of verb-object, action-noun and other than verb-object/action-noun labels in the set of the process activity labels $Labels$. $0 < Ls(P) \leq 1$; $Ls(P)$ is 1 if no label in the style (vo)/(an) exists in the process. We evaluate the style category of labels by exploiting the MINIPAR[11] linguistic analyser. We expect that a high value of the metric indicates a high ratio of ambiguous labels. For instance, in the $PM_{example1}$, two labels are in the (vo) form ("Process Order" and "Collect Item"), two in the (an) form ("Item Retrieval" and "Order Closure") and one in the (rest) style ("I.S.P"), hence $Ls(PM_{example1}) = 0.67$.

- ***Structure size*** (Sz). It refers to the number of elements, in terms of activities and gateways, contained in the process model [28].

$$Sz(P) = |Activities| + |Gateways| + |Events| = |FlowObjects|$$

where $Activities$, $Gateways$, and $Events$ are respectively the sets of activities, gateways and events of the process, i.e., the set of process flow objects. $Sz(P) \geq 0$; $Sz(P)$ is 0 if the process is empty. We expect that a high value of the metric indicates a large process. For instance, $PM_{example1}$ contains 5 activities and 2 gateways, hence, $Sz(PM_{example1}) = 7$.

- ***Density*** (Sd). It refers to the density of connections between pairs of process activities and/or gateways [180].

$$Sd(P) = \frac{|SequenceFlows|}{|FlowObjects|*(|FlowObjects|-1)}$$

where $SequenceFlows$ and $FlowObjects$ are respectively the set of sequence flows and flow objects (i.e., activities, gateways and events) in the process. $0 \leq Sd(p) \leq 1$; $Sd(P)$ is 0 if no flow exists among process flow objects, 1 if in the process each flow object is connected to any other flow object. We expect that a high value of the metric implies a high connectivity. For instance, $PM_{example1}$ contains 7 flow objects (5 activities and 2 gateways) and 7 sequence flows, hence, $Sd(PM_{example1}) = 0.16$

- ***Net Complexity Coefficient*** ($Scnc$). Similarly to Sd, it refers to the relation between connections (sequence flows) and flow objects (activities, gateways and events) of the process model [28].

[11] `http://www.cs.ualberta.ca/~lindek/minipar.htm`.

$$Scnc(P) = \frac{|SequenceFlows|}{|FlowObjects|}$$

$0 \leq Scnc(P) \leq |FlowObjects| - 1$; $Scnc(P)$ is 0 if no flow exists among process flow objects, while its maximum value corresponds to the number of process flow objects when in the process each flow object is connected to any other flow object. We expect that a high value of the metric indicates a complex control flow. For instance, $PM_{example1}$ contains 7 flow objects (5 activities and 2 gateways) and 7 sequence flows, hence, $Scnc(PM_{example1}) = 1$

- ***Control-Flow Complexity*** ($Scfc$). It refers to the number of alternative execution flows contained in the whole process [28].

$$\begin{aligned} Scfc(P) \quad = \quad & \sum_{g \in P \wedge g \in Gateways_{decision_{AND}}} Scfc_{AND}(g) + \\ & \sum_{g \in P \wedge g \in Gateways_{decision_{XOR}}} Scfc_{XOR}(g) + \\ & \sum_{g \in P \wedge g \in Gateways_{decision_{OR}}} Scfc_{OR}(g) \end{aligned}$$

where $Gateways_{decision_{AND}}$, $Gateways_{decision_{XOR}}$ and $Gateways_{decision_{OR}}$ are the set of process gateways representing decision points of type AND, XOR and OR, respectively; g is one of these points in the process; and $Scfc$ of a decision gateway g varies according to the number of different process states that can be reached once the gateway of the specific type has been executed. A process state is the description of the state of the process in a precise time (i.e., the set of activities that can be executed in that precise instant), corresponding to the mental state of a designer. Hence, $Scfc_{AND}(g) = 1$ (since a unique state containing all the activities outgoing from g can be activated); $Scfc_{XOR}(g) = FanOut(g)$, where $FanOut(g)$ is the number of sequence flows in output to the decision point g (since FanOut possible states can be activated); and $Scfc_{OR}(g) = 2^{FanOut(g)} - 1$ (since $2^{FanOut(g)} - 1$ states can be activated with $FanOut$ outgoing sequence flows). $Scfc(P) \geq 0$; $Scfc(P)$ is 0 if no decision gateway exists in the process. We expect that a high value of the metric indicates a high complexity of the process flow. For instance, $PM_{example1}$ contains 2 XOR gateways ($g1$ and $g2$) with $FanOut(g1) = 1$ and $FanOut(g2) = 2$, respectively; hence, $Scfc(PM_{example1}) = 3$.

- ***Average Connector Degree*** ($Sacd$). It represents the average number of sequence flows incoming to ($FanIn$) or outgoing from ($FanOut$) process gateways [114].

$$Sacd(P) = \frac{\sum_{g \in Gateways} FanIn(g) + FanOut(g)}{|Gateways|}$$

$Sacd(P) \geq 0$; $Sacd(P)$ is 0 if no decision/merging gateway exists in the process. We expect that a high value of the metric indicates a high complexity of the process flow. For instance, the $PM_{example1}$ model contains 2 gateways ($g1$ and $g2$) with, respectively, $FanOut$ equal to 2 and 1 and $FanIn$ equal to 1 and 2; hence, $Sacd(PM_{example1}) = 3$.

- ***Cross Connectivity*** (Scc). It measures how "strong" is the connection between all the possible pairs of connected process flow objects [181], where, intuitively, the strength of a connection between two flow objects is the degree of freedom from possible alternative choices in the path connecting the two flow objects.

$$Scc(P) = \sum_{fo1,fo2 \in FlowObjects \wedge fo1 \neq fo2 \wedge (\exists flow_{fo1,fo2})} CC(flow_{fo1,fo2})$$

where $\exists flow_{fo1,fo2}$ means that there exists at least one process execution flow (i.e., path in the process) connecting the flow object $fo1$ to the flow object $fo2$ and

$$CC(flow_{fo1,fo2}) = \max_{flow_{fo1,fo2} \in FLOW_{fo1,fo2}} \left(\prod_{sf \in flow_{fo1,fo2}} CC(source(sf)) * CC(target(sf)) \right)$$

where $FLOW_{fo1,fo2}$ is the set of all possible execution flows between $fo1$ and $fo2$. For a generic flow object fo, $CC(fo) = 1$ in case of $fo \in Activity \cup Events \cup Gateways_{AND}$, while $CC(fo) = \frac{1}{d(fo)}$ in case of $fo \in Gateways_{XOR}$ and $CC(fo) = \frac{1}{2^{d(fo)}-1} + \frac{2^{d(fo)}-2}{2^{d(fo)}-1} * \frac{1}{d(fo)}$ in case of $fo \in Gateways_{OR}$, where $Gateways_{AND}$, $Gateways_{XOR}$ and $Gateways_{OR}$ are respectively the set of parallel, exclusive and inclusive gateways and $d(fo) = FanIn(fo) + FanOut(fo)$. The intuition behind this metric is that the strength of a sequence flow between two flow objects varies according to the type of objects it directly connects; for example, a sequence flow directly connecting an AND gateway with an activity is stronger than a sequence flow directly connecting a XOR gateway with an activity, because the latter is an optional sequence flow. $Scc(P) \geq 0$; $Scc(P)$ is 0 if no flow exists among flow objects. There exists also a relative version of this metric that normalizes the absolute metric with respect to the theoretical maximum number of paths between all the flow objects. We expect that a high value of the metric indicates a process flow with few possible alternative choices in the path connecting objects. For instance, in case of $PM_{example1}$, $Scc(PM_{example1}) = 75.6$

- ***Separability*** ($Ssep$). It refers to the degree of separability of a process [114].

$$Ssep(P) = \frac{|CutElements|}{|FlowObjects|}$$

where $CutElements$ represents the set of process flow objects that, if removed from the process, separate the process into disconnected components (i.e., when removing a cut-element from the process, at least one among the process flow objects is disconnected from the process *Start* element). $0 \leq Ssep(P) \leq 1$; $Ssep(P)$ is 1 if all the process flow objects generate disconnected components when removed from the process, 0 if it does not happen for any flow object. We expect that a high value of the metric indicates a well-structured process. For instance, the $PM_{example1}$ contains 2 $CutElements$ (the two gateways of the process), hence, $Ssep(PM_{example1}) = 0.28$.

- ***Sequentiality*** ($Sseq$). It refers to the number of "sequential" sequence flows contained in the process [114]. A sequence flow between two flow objects is "sequential" if it does not connect gateways.

$$Sseq(P) = \frac{|SE_{SequenceFlows}|}{|SequenceFlows|}$$

where $SE_{SequenceFlows}$ represents the set of sequence flows connecting only activities and events (i.e., source and target flow objects of the sequence flow are not gateways). $0 \leq Sseq(P) \leq 1$; $Sseq(P)$ will be 1 if the process does not contain gateways, 0 if no connection exists between pairs of activities. We expect that a high value of the metric indicates a low complexity. For instance, the $PM_{example1}$ contains 2 $SE_{SequenceFlows}$ (i.e., the one connecting B to C and the one connecting C to D), hence, $Sseq(PM_{example1}) = 0.28$.

- ***Depth*** ($Sdep$). It refers to the nesting degree of flow objects in the process [75]. The nesting depth of a process flow object is given by the minimum number of decision elements (not matched by any merging point) that have to be traversed for reaching the flow object itself. *Sdep* of a process represents the maximum value of nesting depth of its flow objects.

$$Sdep(P) = max_{fo \in FlowObjects} \{depth(fo)\}$$

where $depth(fo)$ is the minimum number of decision gateways (not matched by any merging gateway) encountered from the process *Start* element to the flow object fo. $Sdep(P) \geq 0$; $Sdep(P)$ is 0 if no decision gateway exists. We expect that a high value of the metric indicates a highly complex structure of the process. For instance, the *Sdep* of $PM_{example1}$ is equal to 1, since there exists only one level of nesting for reaching activities B, C and D from the *Start* element.

- ***Connector Mismatch*** (Scm). It refers to the mismatch, if any, between the number of sequence flows outgoing from the decision gateways and the number of sequence flows incoming to merging gateways [114]. $Scm(P) \geq 0$; $Scm(P)$ is 0 if, for each sequence flow outgoing from a decision gateway, there exists at least a sequence flow incoming to a merging gateway.

$$\begin{aligned} Scm(P) \;=\; & \left| \sum_{g \in Gateways_{decision_{AND}}} FanOut(g) - \sum_{g \in Gateways_{merging_{AND}}} FanIn(g) \right| + \\ & \left| \sum_{g \in Gateways_{decision_{XOR}}} FanOut(g) - \sum_{g \in Gateways_{merging_{XOR}}} FanIn(g) \right| + \\ & \left| \sum_{g \in Gateways_{decision_{OR}}} FanOut(g) - \sum_{g \in Gateways_{merging_{OR}}} FanIn(g) \right| \end{aligned}$$

We expect that a high value of the metric indicates a non well-structured process. For instance, considering both FanOut and FanIn of the two gateways of $PM_{example1}$, $Scm(PM_{example1}) = 0$.

- ***Cyclicity*** (Scy). It measures the number of process flow objects involved in cyclic process flows (i.e., cyclic connection among elements) [114].

$$Scy(P) = \frac{|ElementsInCycles|}{|FlowObjects|}$$

where $ElementsInCycles$ represents the set of flow objects contained in a cycle. $0 \leq Scy(P) \leq 1$; $Scy(P)$ is 1 if all the considered process elements are involved in a cycle, 0 if there are not cyclic flows. We expect that a high value of the metric indicates the presence of a high number of process flow objects in cycles, hence, we expect a complex process control flow. For instance, 3 activities and 2 gateways out of 7 elements of the process $PM_{example}$ are involved in a cycle, hence, $Scy(PM_{example1}) = 0.71$.

- ***FanIn/FanOut*** ($Sfifo$). It refers to the connection degree between external flow objects and sub-processes [28, 75].

$$Sfifo(P) = \frac{\sum_{subp \in Subprocesses} Sfifo(subp)}{|Subprocesses|}$$

where $Subprocesses$ is the set of process sub-processes, *subp* is one of them and $Sfifo(subp)$ measures the connection between *subp* and external flow objects. The flow object interconnection of *subp*, $Sfifo(subp)$, i.e., the complexity of the connections of *subp* with its environment, is computed by considering the quadratic product of the sub-process FanIn and FanOut, $Sfifo(subp) = (FanIn(subp) * FanOut(subp))^2$, where $FanIn(subp) * FanOut(subp)$ represents the total number of possible combinations of a *subp* incoming sequence flow to a *subp* outgoing sequence flow. $Sfifo(P) \geq 0$; $Sfifo(P)$ is 0 if no sub-process exists in the process. We expect that a high value of the metric indicates a highly complex structural process. For instance, $Sfifo$ is 0 for the process $PM_{example1}$ since no

sub-process is in the model. $Sfifo$ is 1 in the clustered version of the same process model, $PM_{example2}$, in which the only sub-process has both fan-in and fan-out equal to 1.
- ***Structural appropriateness*** (Ac). It refers to the structural complexity of a process model in terms of the proportion of flow objects that are not control flow objects with respect to the whole set of process flow objects [155].

$$Ac(P) = \frac{|Activities|+|Events|}{|FlowObjects|}$$

$0 < Ac(P) \leq 1$; $Ac(P)$ is 1 if the process does not contain gateways. We expect that a high value of the metric indicates a high process model conformance (i.e., a limited model generalization). For instance, $PM_{example1}$ contains 5 activities and 2 gateways, hence, $Ac(PM_{example1}) = 0.71$.
- ***Behavioural flexibility*** (Ab). It refers to the model flexibility [155]. It takes into account both the flexibility introduced by the different behaviours described in the original artefacts used to recover the process model (e.g., in case of dynamic analysis, log files) and the generalization introduced by the model with respect to the artefacts themselves. In particular, it measures the average number of behaviours added by the model with respect to the model listing all the possible sequences described in the initial artefacts, hence including also behaviours not traced in the considered artefacts. To compute this metric, the behaviours directly documented in the artefacts (e.g., traces) are "re-parsed" and reproduced in the generated process model, thus computing the average quantity of additional behaviours allowed by the model for each traversed node. In the existing literature, Ab is only defined to be measured for process models recovered from execution traces. Since we use a dynamic approach from log files for process model recovery, we preserve such a definition but we are aware that, in case of recovery from different initial artefacts (e.g., behaviours described in documentation scenarios, UML sequence diagrams), it needs to be extended.

$$Ab(P) = \frac{\sum_{tr \in Traces}(|Activities \cup Events| - Z(tr))}{|Traces| * (|Activities \cup Events| - 1)}$$

where $Z(tr)$ measures the average number of (direct or passing through gateways) connections between process activities or events, traversed when re-parsing the trace tr, and activities or events, i.e., $Z(tr) = \frac{\sum_{i=1}^{ne(tr)-1} noSF(i)}{ne(tr)-1}$, where $ne(tr)$ is the number of activities or events collected in the trace tr and $noSF(i)$ is the number of (direct or passing through gateways) connections from the process activity or event i, traversed when reproducing the trace tr, and another process activity or event (or the process element itself if it is cyclic). $0 \leq Ab(P) \leq 1$; $Ab(P)$ is 1 if the process does not contain gateways while it is 0 if all the actvities and events can cycle on themselves and all the pairs of distinct activities and

events are connected. We expect that a high value of the metric indicates a limited generalization introduced in the process model with respect to the initial traces. For instance, considering the three traces *t1*, *t2* and *t3* in Figure 3.14, $Ab(PM_{example1}) = 0.89$, since $|Activities| = 5$, $|Traces| = 3$, $Z(t1) = \frac{6}{4} = 1.5$, $Z(t2) = \frac{6}{4} = 1.5$ and $Z(t3) = \frac{9}{7} = 1.29$.
- ***Fitness*** ($Afit$). It refers to the "fitting" degree between the initial artefacts (e.g., traces) and the generated process model [155]. In other terms, it estimates the amount of behaviour documented in the initial artefacts that is reproduced by the process model. Also in this case, as for the *Ab* metric, we refer to $Afit$ as a metric devoted to measure the conformity between process models and traces. To compute $Afit$, the traces are re-parsed for traversing the process model and the identified mismatches are considered.

$$Afit(P) = 1 - \frac{1}{|Traces|} * \sum_{tr \in Traces} \frac{reqSF_{tr}}{exsSF_{tr}}$$

where $reqSF_{tr}$ is the number of sequence flows to be added to the process model to make it able to correctly reproduce the trace tr, and $exsSF_{tr}$ the number of sequence flows (including those added to the model for reproducing tr) that are activated when reproducing the trace tr in the process. $0 \leq Afit(P) \leq 1$; $Afit(P)$ is 1 if the process model captures all the behaviours described in the initial traces, 0 if none of them is captured. We expect that a high value of the metric indicates a process model that completely describes the traces and thus is, often, rich and complex. A low value of $Afit$ may indicate that the model is not sufficiently adequate to describe the actual behaviours of the process. For instance, considering $t1$, $t2$ and $t3$, $Afit(PM_{example}) = 1$ since $reqSF = 0$, for each of the three traces.

3.2.2 Experimental Study

Following the indications in the related literature (e.g., Briand et al. [1], Rolon et al. [55] and Alves et al. [168]), a two-step pilot experimental study has been conducted to test the validity of the proposed metrics for reverse engineered process models. In the first step, an experiment with human modellers has been performed for evaluating the connection between the proposed suite of metrics and the process model understandability. In the second step, another experiment based on the construction and evaluation of a prediction model has been conducted to evaluate the use of such metrics for estimating the understandability in absence of human modellers.

Seven e-commerce applications (Softslate, Erol[12], Communicart[13], Avactis[14], Inter-

[12]`http://www.eroldemostore.co.uk`
[13]`http://www.communicart.biz`
[14]`http://www.avactis.com`

spire[15], Digistore[16], and WinestoreEvol[17]) have been selected and analysed to infer the processes they implement (by using the model recovery technique described in Subsection 3.1). All of them represent medium/large (in terms of application size and complexity) Web applications implementing shopping carts for on-line stores. They realize functionalities to support the on-line retail of products (e.g., catalog, cart, order form and payment checkout management) and systems for handling customer accounts as well as product shipping. Six applications out of seven are real e-commerce systems (three of them have already been introduced in Subsection 3.1.7 for the evaluation of the reverse engineering technique) widely used and adopted to implement on-line stores and do real business. WinestoreEvol, instead, represents a realistic e-commerce application implemented by students at FBK. It evolves an application published in a book [193] as example of PHP language. It was also used as case study in other works (e.g., [108]).

Metrics Distribution and Correlation Analysis

The first step of the experiment has been conducted to empirically assess the connection between metrics and process understandability. In detail, the main aim of the experiment was to answer the following research question:

RQ1 *Is there correlation between each metric of our suite, considered in isolation, and the process model understandability?*

Objects
Our process model recovery technique has been applied to the seven Web applications by using different settings (e.g., without the clustering-based refinement, with different combinations of the clustering techniques, with different sets of traces), thus obtaining different recovered process models for each application. Overall, we generated 35 process models (5 processes for each Web application).
Subjects
Six human process modellers have been involved in the experiment with the aim of collecting their opinion about the understandability of the recovered models. All the modellers are Ph.D. or Ph.D. students working with processes at FBK.
Variables
The process model measurements obtained with our metrics represent the independent

[15]`http://www.interspire.com`
[16]`http://www.digistore.it`
[17]`http://www.webdatabasebook.com`

WAs:	A, B, C				
BPs:	p1 (flat), p2, p3, p4, p5 (clustered)				
Experts:	e1, e2, e3				
	Lab1	Lab2	Lab3	Lab4	Lab5
	A_p1	A_p2	A_p3	A_p4	A_p5
	B_p5	B_p4	B_p3	B_p2	B_p1
	C_p2	C_p5	C_p1	C_p3	C_p4

Table 3.3: Experiment design

Structure											
	Sz	Sd	$Scnc$	$Scfc$	$Sacd$	Scc	$Ssep$	$Sdep$	Scy	Scm	$Sfifo$
min	22	0.001	1.09	18	4.42	0.80	0.31	0	0	4.00	0.30
max	152	0.07	2.76	168	11.20	18.90	0.99	7	0.73	109.00	1.00
avg	57.37	0.03	1.47	62.03	5.78	11.31	0.72	4.17	0.34	32.80	0.71
median	43.50	0.03	1.43	50.50	5.30	11.00	0.78	4.50	0.30	19.50	0.63
st.dev.	34.36	0.02	0.30	38.35	1.70	3.75	0.28	1.80	0.22	30.25	0.24

Term			
	Ll	$L0d$	Ls
min	0.15	0.61	0.43
max	1.00	0.98	0.89
avg	0.76	0.85	0.66
median	0.94	0.88	0.64
st.dev.	0.29	0.10	0.10

Conformance			
	$Afit$	Ab	Ac
min	2.00	0.30	0.47
max	3.96	1.50	1.56
avg	2.92	0.83	0.89
median	3.05	0.71	0.86
st.dev.	0.59	0.37	0.35

Table 3.4: Descriptive statistics of the metric measures, organized according to the corresponding process understandability factor, obtained for the 35 process models involved in the experiment. Note that $Sseq$ is not reported since in the considered recovered models it is always 0.

Structure											
	Sz	Sd	$Scnc$	$Scfc$	$Sacd$	Scc	$Ssep$	$Sdep$	Scy	Scm	$Sfifo$
min	0.14	0.02	0.39	0.11	0.39	0.04	0.31	0.00	0.00	0.04	0.30
max	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
avg	0.38	0.46	0.53	0.37	0.52	0.60	0.72	0.60	0.47	0.30	0.71
med.	0.29	0.49	0.52	0.30	0.47	0.58	0.79	0.64	0.41	0.18	0.63
std	0.22	0.24	0.11	0.23	0.15	0.20	0.28	0.25	0.30	0.27	0.24

Term			
	Ll	$L0d$	Ls
min	0.51	0.20	0.30
max	1.00	1.00	1.00
avg	0.74	0.55	0.57
median	0.77	0.48	0.55
std	0.15	0.24	0.22

Conformance			
	$Afit$	Ab	Ac
min	0.15	0.62	0.48
max	1.00	1.00	1.00
avg	0.76	0.87	0.74
median	0.94	0.90	0.72
std.	0.29	0.10	0.11

Table 3.5: Descriptive statistics of the linearly normalized metric measures, organized according to the corresponding process understandability factor, obtained for the 35 process models involved in the experiment. Note that $Sseq$ is not reported since in the considered recovered models it is always 0.

variables in the experiment, while the process model understandability, defined according to the modellers' opinions, the dependent variables.

Design

We considered six out of the seven Web applications involved in the experiment (i.e., Digistore was excluded in this experiment), randomly divided into two sets, each one used in one of the two experiment iterations. The experiment has been repeated two times considering each time a different set of Web applications (Softslate, Erol and Communicart in the first iteration and Avactis, Interspire and WinestoreEvol in the second one) and of human modellers (three in the first iteration and three in the second one)[18]. For each experiment iteration, we adopted a simple experiment design [194] intended to fit a set of five 1-hour lab sessions in order to limit the learning effect on applications. Each modeller was asked to analyse and evaluate more models of the same application. Table 3.3 summarizes the experiment design: we extracted five process models (*p1* to *p5*) from each of the three Web applications considered (A, B, and C) and we involved three modellers (*e1*, *e2* and *e3*). In detail, *p1* is the "flat" process model obtained without applying the clustering-based process refinement, while *p5* represents the clustered version of the model; the remaining processes (*p2*, *p3* and *p4*) are obtained by applying different combinations of the clustering techniques, with different sets of traces. The distribution of the processes to be evaluated in the lab sessions, their combination, and the order in which they are provided to the modellers have been chosen for limiting the learning effect and the impact of co-founding factors. For instance, in the first lab session, the experts have been provided with three models inferred from three different Web applications: a flat model (A_p1), a clustered model (B_p5) and a third process model with a random recovery technique setting and configuration (C_p2).

Procedure

For each of the two iterations, we performed the following steps:

1 For each of the three Web applications, we applied the model recovery technique described in Section 3.1 by considering five different settings and technique configurations. In detail, we applied the recovering technique (i) without the cluster-based modularization, (ii) with the cluster-based modularization considering different combinations of clustering (e.g., sequence-alternative-loop and loop-sequence) and (iii) by considering different sets of software execution traces. The output of this step is a set of five different process models, each representing a different view of the process underlying the considered application (see Figure 3.13 for an example

[18]Both the Web application and the modeller sets have been defined randomly.

of recovered process model).

2 We applied the whole suite of metrics to each recovered process model.

3 We provided the human modellers with five process models (according to the schema in Table 3.3) and we asked them to rate the process model understandability based on their knowledge and experience in process modelling. The modellers' rate about the process model (i.e., the "modellers' opinion" MO) could vary on a 5-point scale (1 = very low, 2 = low, 3 = medium , 4 = high, 5 = very high). Moreover, to support modellers in the understandability evaluation, we asked them to answer five comprehension questions about the process model (e.g., "Is the login operation always required in order to load items in the shopping cart?", "Can the user remove an item from the cart if no item has ever been added to the cart before?") before rating each model. We collected information about the number of correct answers and the time required for answering, though the main aim of the questionnaire was supporting the experts in the process understandability evaluation, hence only the expert opinions MOs have been considered in the experiment.

4 We analysed the collected data by means of four steps: (i) analysis of the descriptive statistics and of the boxplots of the obtained measures, in order to evaluate the trends of the measures; (ii) correlation analysis by means of the Spearman's coefficient between pairs of process metrics, in order to detect unexpected relationships; (3) correlation analysis by means of the Spearman's coefficient between each process metric and the modellers' opinions (MOs), in order to find common trends; and (iv) analysis by means of the Spearman's coefficient and the Wilcoxon statistical test of the potential relationship existing between pairs of opinions expressed by modellers evaluating the same recovered process model. This last step is particularly relevant to evaluate the variability of modellers' subjective opinions about the model understandability.

Summarizing, by considering both the experiment iterations, we analysed six Web applications and recovered, for each of them, five process models representing different views of the process it implements. We then collected the process metrics described in the previous subsection (Subsection 3.2.1) on each generated process model, as well as the opinions of the six modellers about the understandability of each process model. Finally, we analysed the correlation between the collected measures to detect whether relationships exist in their trends.

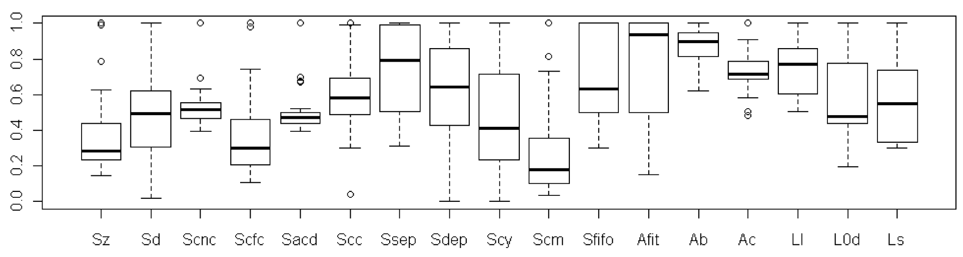


Figure 3.15: Distribution of the measured metrics with respect to the analysed process model (the metric values are linearly normalized).

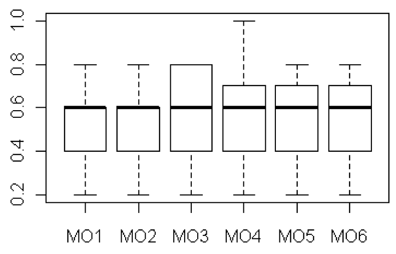


Figure 3.16: Distribution of the modellers' opinions (MOs) with respect to the process understandability (the metric values are linearly normalized).

	MO1	MO2	MO3	MO4	MO5	MO6
min	1	1	1	1	1	1
max	4	4	4	5	4	4
avg	2.53	2.60	2.93	2.73	2.67	2.80
median	3	3	3	3	3	3
st.dev.	1.06	1.06	1.16	1.16	1.11	1.01
min	0.20	0.20	0.20	0.20	0.20	0.20
max	1.00	1.00	0.80	1.00	1.00	1.00
avg	0.51	0.52	0.59	0.55	0.53	0.56
med.	0.60	0.60	0.60	0.60	0.60	0.60
std	0.21	0.21	0.23	0.23	0.22	0.20
	ANS1	ANS2	ANS3	ANS4	ANS5	ANS6
avg	98.6	96	97.3	89.3	97.6	92.1
	T1	T2	T3	T4	T5	T6
avg	378.6	360	450	437.5	510.4	401.3

Table 3.6: Descriptive statistics about the (non-normalized and normalized) modellers' understandability opinions MOs, percentage of correct answers ANS, and required time T (sec.) for a set of 30 recovered processes

Results: Process Metrics and Understandability correlation

Table 3.5 reports the descriptive statistics of the obtained process measurements (the non-normalized values are reported in Table 3.4), while Figure 3.15 presents the boxplots of these measures. Boxplots show that, for some metrics (e.g., *Sd*, *Sdep*), values are widely distributed in the range [0-1], differently from other metrics (e.g., the values of *Scnc* and *Sacd* are concentrated around 0.5 while *Ab* and *Ac* around 0.8).

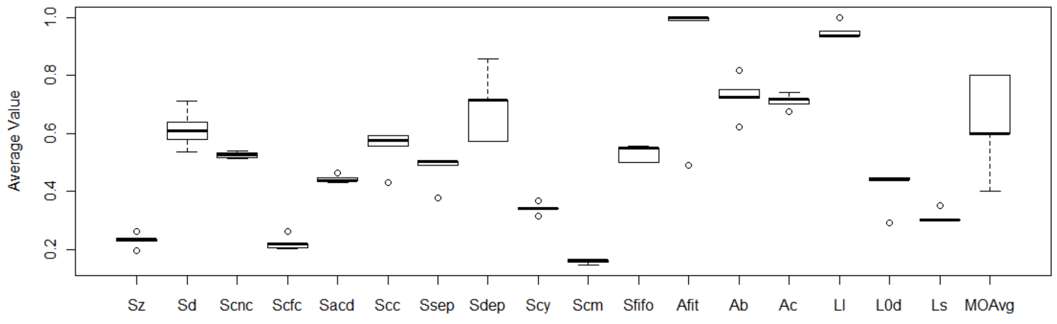
Table 3.6 reports the descriptive statistics related to the non-normalized and normalized modellers' opinions (MOs) about the process understandability (the corresponding boxplots are depicted in Figure 3.16), as well as, the average percentage of correct answers ANS and the average time T (expressed in seconds) required to complete the comprehension questionnaire provided to subjects for helping them in expressing their evaluation on the process model understandability. The table shows that the percentage of correct answers given by modellers is high (on average 95%) and the time spent is quite limited (on average 1.4 minutes for question). For completeness, we reported all the collected data in Table 3.6, however, we recall that only the expert opinions MOs have been considered in the experiment, while the purpose of the questionnaire was helping experts in the evaluation of process understandability.

Figure 3.17 reports instead, individually, the boxplots of both normalized metrics and modellers' opinions collected for the applications evaluated by the first group of subjects: Softslate, Communicart and Erol. The boxplots give an idea of how metrics and average modellers' opinion vary for each application as well as of the differences in the distribution of metrics and modellers' opinions among applications. In particular, Erol has been perceived by the first group of subjects as the most difficult application to understand.

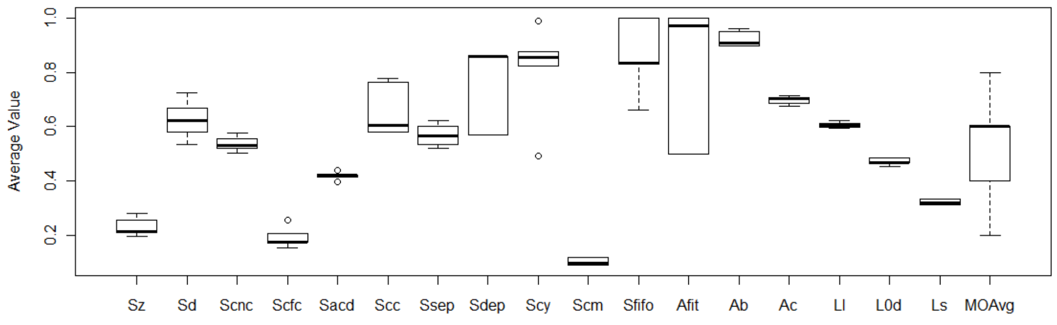
By plotting the process model metrics versus the experts' opinions, in most of the cases, clear linear and curvilinear relationships come out. We hence used the Spearman's correlation coefficient to evaluate these relationships[19]. We computed correlation on normalized metric values; since a linear normalization is applied, the value of the correlation is not affected. Table 3.7 shows the correlation results. The table reports only the results that are statistically relevant at 5 percent confidence level, i.e., there is the 5% of probability that the correlation is coincidental (p-value$\leq$0.05). Data reported in the table show that Sz and $Scfc$ have a strong (negative) correlation[20] with the modellers' understandability evaluation, while Sd and $Afit$ have a moderate correlation. This outcome

[19]The correlation coefficient ranges from -1 (a negative correlation) to 1 (a positive correlation), while a coefficient of 0 means no correlation.

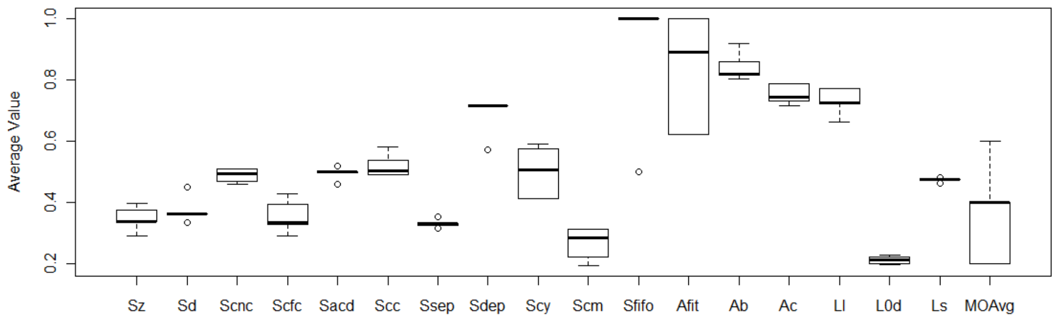
[20]The use of terms "strong" and "moderate" for describing the obtained statistical correlation is based on a well-known convention [35].



(a) Softslate

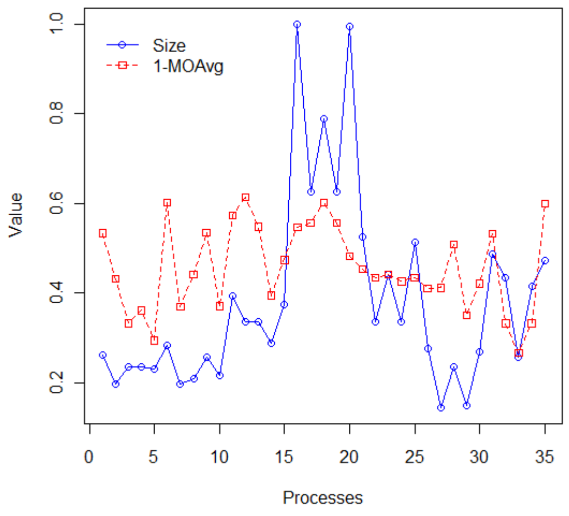


(b) Communicart

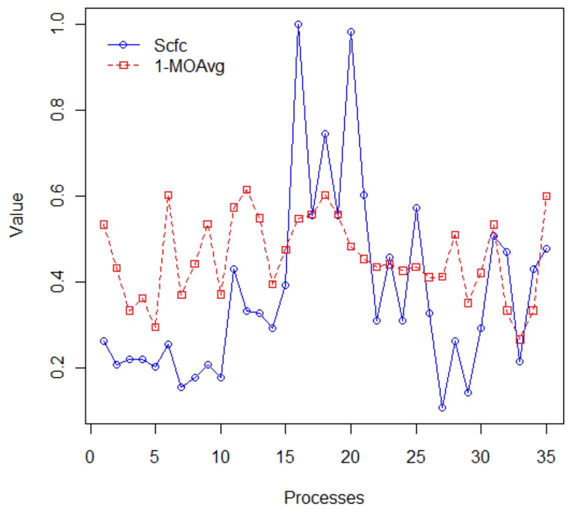


(c) Erol

Figure 3.17: Softslate, Communicart and Erol boxplots



(a) Sz vs. 1-MOAvg



(b) Scfc vs. 1-MOAvg

Figure 3.18: Plot of some metrics and modellers' opinions

seems consistent with the results obtained in previous works (e.g., [110] and [28]) and confirmed by Figure 3.18, reporting the plots of Sz versus $1 - MOAvg$ and $Scfc$ versus $1 - MOAvg$. Plots have similar shapes, though variations in average modellers' opinion are limited when compared to Sz's and $Scfc$'s peaks. Other metrics (i.e., *Sacd*, *Scy*, *Scm* and *L0d*), instead, show some degree of correlation with the modellers' opinions while only non-statistically relevant results have been observed for the metrics not listed in Table 3.7. According to the obtained overall results, hence, we can empirically validate 8 of the proposed metrics (i.e., *Sz*, *Sd*, *Scfc*, *Sacd*, *Scy*, *Scm*, *Afit* and *L0d*) with respect to the research question **RQ1**, by assessing their connection with the process model understandability. On the contrary, we cannot positively answer the research question **RQ1**, in case of the remaining metrics.

Beyond the information related to the specific metrics, results in Table 3.7 also suggest interesting observations about the relationship between process model properties and understandability. According to the table, in fact, process size (measured by Sz) and flow (measured by *Sd* and $Scfc$) are the properties more related to the process understandability. On the contrary, metrics depending on more properties of the process model (e.g., *Scnc* depends on the number of both flow objects and sequence flows, i.e., on both process size and process flow), do not have statistically relevant correlations with

Metric	Corr.(%)	p-value
Sz	-59.1	$8.5e^{-10}$
$Scfc$	-58.3	$1.6e^{-9}$
Sd	54	$3.4e^{-8}$
$Afit$	-52.2	$1.27e^{-7}$
$Sacd$	-46.9	$3e^{-6}$
Scm	-38	0.0002
Scy	-21.3	0.043
$L0d$	19.9	0.05

Table 3.7: Spearman's correlation between process metrics and modellers' opinions on the understandability. The table shows only the metrics with a $p - value \leq 0.05$.

the process model understandability. Nevertheless, this result could be due to the limited variance of this type of metrics (e.g., the standard deviation of $Scnc$ is very low, as shown in Table 3.5 and in the boxplot in Figure 3.15), thus suggesting the need to perform further investigations devoted to study how a higher variance of these metrics can affect the obtained results. Finally, further investigations would also be advisable for the three metrics related to the quality of the activity labels. In fact, most of the labels are single words or very short sentences rather than complete-sense sentences (the value of the Ll metric is in general quite low, as shown in Table 3.5). This result is reasonable since labels are automatically recovered during the process inference. Moreover, the use of few words for composing activity labels influences negatively the label style and hence model understandability (in fact, the value of the Ls metric is negatively correlated with the understandability). Probably, a user intervention during the label recovery or a label-refinement step could be required to better assess the impact of labels in model understandability.

Tables 3.8 summarizes the correlation existing between pairs of metrics measuring the same factor according to the measurements collected in the recovered process models. Tables report both the statistically relevant ($p - value < 0.05$) and non-relevant values.

As expected, limited/moderate correlations exist between pairs of metrics in label and conformance categories, while some strong correlations exist between pairs of process structure metrics. This outcome was expected since, in our suite, different metrics have been proposed to measure the same process property by considering different granularities and points of view. For instance, Sz (size), Sd (density) and $Scfc$ (control flow complexity) strongly correlate: Sz positively correlates with $Scfc$, while both negatively correlate with Sd. The reason is that, in a process with a non-trivial control flow (e.g., the flow is neither a chain of activities and events nor it describes a process in which each

	Sz	Sd	$Scnc$	$Scfc$	$Sacd$	Scc	$Ssep$	$Sdep$	Scy	Scm	$Sfifo$
Sz	1	-97.1	-33.2	97.5	82.4	32.3	48.7	3.3*	-15.2*	89.1	-22.4
Sd		1	38.1	-93.5	-77.9	-36.4	-44*	-16.1	20.4*	-91.8	24.5
$Scnc$			1	-24.7	-44	30.4	-1.8	-0.7*	63	-47.7	38.1
$Scfc$				1	82.4	29.1*	48.4	-1.3*	-12.4*	86	-22.7
$Sacd$					1	-6.7*	35.8	-11.7	-45.1	79.7	-40.4
Scc						1	35.5	43.1	39.4	23.1	6.8*
$Ssep$							1	-19.8*	-24.1	39.2	31.6
$Sdep$								1	23.6	16.1*	12.5*
Scy									1	-34.2	52.8
Scm										1	-39.4
$Sfifo$											1

	Ll	$L0d$	Ls
Ll	1	-25.1	-39.8
$L0d$		1	70.3
Ls			1

	$Afit$	Ab	Ac
$Afit$	1	-56.5	-9.1*
Ab		1	36.4
Ac			1

Table 3.8: Spearman's correlation (%) between pairs of structure (top table), label (middle table) and conformace (bottom table) process metrics (* indicates correlations with $p - value > 0.05$)

flow object is connected to any other), a non-trivial increase of the number of flow objects (i.e., Sz) likely implies a consistent increase of sequence flows in input to/output from the process gateways[21] (i.e., $Scfc$), thus slightly varying their ratio. Hence, this consistent increase (of both flow objects and sequence flows) reasonably determines that the ratio between the number of sequence flows and the number of all the potential sequence flows connecting each process flow object to any other (i.e., about the square of the process size) decreases. Such a relation of Sz (size) with Sd and $Scfc$, confirms hence the results reported in Table 3.5.

Summarizing, Table 3.8 shows that some metrics in our suite can be considered surrogates of other metrics since similar and correlated trends have been observed.

Finally, to evaluate the agreement between modellers, we applied the Spearman's correlation coefficient to the understandability opinions MOs expressed by different modellers on the same process. According to the experiment design, in fact, each process model has been evaluated by three human modellers. Table 3.9 shows that the correlations between pairs of modellers' opinions on the same process are very high (on average 83.4%) and the results are always statistically significative. We additionally perfomed a Wilcoxon statistical test with the aim of investigating whether the opinions expressed by the three modellers (i.e., the three experts involved in the first and in the second iteration of the experiment, respectively) about the understandability of the same set of models were consistent. In detail, we tried to answer the following question:

Do the opinions we obtained from the tree experts on the same process have the same values?

According to this question we formulated the following null-hypothesis: (H0) *The modellers' opinions on the same process do not diverge* and the alternative hypothesis: (Ha) *Modellers' opinions diverge.* Table 3.9 shows the obtained p-values. For none of the pairs of modellers' opinions we were able to reject the null hypothesis since the obtained p-value is always $p - value > \alpha_{Bonferroni}$ ($\alpha_{Bonferroni} = 0.0083$). In fact, since we used a repeated statistical test, the Bonferroni correction has to be applied (i.e., the null hypothesis can be rejected if and only if the p-values are lower than $\alpha_{Bonferroni} = 0.05/6 = 0.0083$). Both the results obtained in the correlation analysis and in the Wilcoxon test show that strong consistency exists among the modellers' evaluations. This outcome corroborates the validation of the obtained results even in case, as in this experiment, the number of subjects is limited.

[21]We assume activities can have only one input and output sequence flow.

	Corr.(%)	p-value	Wilcoxon p-value
MO1-MO2	85.5	$4.8e^{-5}$	1
MO1-MO3	86.4	$3.2e^{-5}$	0.03
MO2-MO3	72.5	0.0022	0.09
MO4-MO5	85.5	0.0002	0.58
MO4-MO6	85.5	0.0006	0.77
MO5-MO6	85.5	$1.1e^{-8}$	0.37

Table 3.9: Analysis of the modellers' opinions

Prediction Analysis

In the second step of the experiment, the metrics collected for the recovered processes have been used in a prediction system for assessing their ability in predicting model understandability. This result helps in evaluating the suitability of the metrics as early understandability predictors for recovered process models. In detail, the main aim of this part of the experiment is answering the following research question:

RQ2 *Are the metrics of our suite predictors of process model understandability?*

To this purpose, we built a prediction model that, using the metrics computed for the considered process models, predicts the modellers' opinions and we checked the obtained predictions against the actual modellers' opinions. In detail, the following steps have been performed:

1 Process metrics reduction by applying the Principal Component Analysis (PCA);

2 For different disjoint sets $data_{model}$ and $data_{test}$ from the initial data set:

   (a) Prediction model construction by applying a stepwise linear regression model to $data_{model}$ set;

   (b) Prediction model evaluation by measuring the accuracy of the prediction. The accuracy is computed by comparing the predicted and the actual (i.e., provided by the human modeller) understandability values for each process model in the $data_{test}$ set.

In the first step the Principal Component Analysis (PCA) has been applied in order to understand whether the initial set of process metrics in the suite can be reduced by removing some related metrics. PCA is a statistical test that reduces complex sets of data composed of many dimensions to smaller ones.

In the second step, more prediction models have been built and evaluated by using different $data_{model}$ and $data_{test}$ sets, in order to validate the proposed metrics as predictors of the process model understandability.

In detail, for each considered pair of $data_{model}$ and $data_{test}$, a (forward) stepwise linear regression analysis [87] has been performed with the aim of understanding how process model understandability (dependent variable) varies when one of the measurements (independent variables) in the $data_{model}$ set of observations changes and the others are kept fixed. By means of the forward stepwise regression analysis, a linear regression model is iteratively built considering different sets of the independent variables for each iteration. The selection of the variable to be added to the model in each iteration is performed by evaluating each variable according to its statistical relevance with respect to the variance of the dependent variable. Hence, in each iteration, the independent variable that captures more variation of the dependent variable is selected for being added to the model. At the end, the obtained regression model is composed of the subset of the independent variables that explains the maximum variance of the dependent variable. Such a regression model is also used for the prediction of the value of the dependent variable.

For the evaluation of the prediction models, two rounds of validation have been applied.

In the first round, we used the leave-one-out cross-validation strategy (Myrtveit et al [127]) in which the prediction model is built by removing, on a rotating basis, one instance from the set of (N) observations in the data set used for building the prediction model and using it as a test case ($data_{test}$ set) for evaluating the prediction. In other terms, only the remaining ($N - 1$) observations are used to generate the model aimed at predicting the value of the removed case (i.e., as $data_{model}$ set). In general, the leave-one-out cross-validation consists in dividing the initial data in k parts and repeating the prediction, using $k - 1$ parts as dataset ($data_{model}$ set) and one for the prediction test ($data_{test}$ set), by rotating the part used as prediction test over all the k parts. By averaging the validation results over the rounds, this analysis provides an evaluation about how the considered metrics generalize to an independent data set. Using a cross-validation is a common procedure (e.g., Myrtveit et al. [127] and Zhou et al. [196]) for evaluating prediction systems when a limited number of observations is available. In our case, in all the leave-one-out cross-validation rounds we used all the processes related to one Web application as $data_{test}$ set for the evaluation of the prediction model and all the process models of the remaining five Web applications as $data_{model}$ set.

In the second round of validation, we built the prediction model by considering the whole set of process metrics of the six considered applications and we evaluated the model using a new set of processes recovered for Digistore (i.e., the Web application not considered for the prediction model construction). In other terms, we generated the prediction model with the process metrics of the 6 applications, and we applied such a model to the five processes recovered for Digistore and evaluated by all the six human modellers. Hence we evaluated the accuracy of the prediction by comparing the actual understandability with the predicted one. This last prediction model takes advantage of a larger set of data (the process models related to all the six applications), and should hence confirm and improve the previous results.

Each built prediction model has been evaluated by comparing the predicted and the actual (i.e., provided by humans) understandability values (of the $data_{test}$ set). To this purpose, we resorted to two metrics used in prediction model evaluation: the means of the absolute error MAE (e.g., [60] and [86]) and $Pred(q)$, i.e., the percentage of predictions falling within q percent of the actual value (e.g., [36], [86] and [196]). They are different measures with different evaluation capabilities: while the first is a measure of error, the second is a measure of accuracy.

In detail, MAE measures the mean of the absolute error of the modellers' opinions predictions (approximated to the closest value) with respect to their actual opinions:

$$MAE = \frac{\sum_{i \in N} |actual(MO_i) - predicted(MO_i)|}{N}$$

where N is the number of observations in the $data_{test}$ (i.e., used in the prediction evaluation), $actual(MO_i)$ and $predicted(MO_i)$ are the i^{th} actual modeller's opinion and the predicted value of the i^{th} modeller's opinion (approximated to the closest value in the Likert scale) on the understandability of the process model, respectively.

$Pred(q)$, instead, measures the percentage of predictions about modellers' opinions lying within $\pm q\%$ of the actual modellers' opinions:

$$Pred(q) = \frac{\left\{ i \in N | \frac{|actual(MO_i) - predicted(MO_i)|}{actual(MO_i)} \leq \frac{q}{100} \right\}}{N}$$

We adapted this measure to our range of values by computing the percentage of predictions of modellers' opinions falling within a range ± 1 on the Likert scale (corresponding to ± 0.2 on the normalized scale), i.e., the percentage of predictions that are either correct or incorrect for at most one point in the Likert scale:

$$Pred_5(q) = \frac{\{i \in N| \ |actual(MO_i) - predicted(MO_i)| \leq q * 0.2\}}{N}$$

In detail, for the prediction model evaluation, we considered MAE and $Pred_5(1)$.

Results: Prediction Model Construction and Evaluation

Applying PCA we obtained a reduction of the initial set of considered metrics. PCA,

Model ID	Metrics	Regression Model	R^2	Adj R
PrM_{all}	all	1.06 - 5.1*Sz+2*Scfc-0.86*Sacd -0.17*Scc+0.54*Sep-0.13*Scy +2.23*Scm+0.46*Ab-0.33*Ls	0.71	0.68
PrM_{pca}	PCA-reduced	0.05496-2.2*Sz+0.4*Sd+0.36*Scnc -0.14*Scc+0.32*Sdep-0.22*Scy +1.8*Scm+0.23*Sfifo+0.42*L0d	0.67	0.63

Table 3.10: Regression models

in fact, suggested the use of 12 out of 18 process metrics (i.e., Sz, Sd, $Scnc$, $Scfc$, Scc, $Sdep$, Scy, Scm, $Sfifo$, $Afit$, Ll, $L0d$). Moreover, this analysis also revealed that five components of PCA allow to explain 86.4% of the total variance.

We applied the (stepwise) linear regression analysis to both the whole set of process metrics measured on the recovered processes and to the reduced set of metrics obtained as output of the PCA. Table 3.10 details the components of both regression models. PrM_{all} is the prediction model built considering all the metrics, while PrM_{pca} the one obtained with the reduced set of metrics. In the table, R^2 represents the amount of variance explained by the model and Adjusted R^2 explains any bias in R^2 by considering the degrees of freedom of the independent variables. By observing the values of R^2, we can conclude that the constructed models can explain a large amount (71% and 67%, respectively) of the total variance. Moreover, we can notice that a limited number of metrics (i.e., Sz, Scc, Scy and Scm) is part of both the regression models. This strengthen the relevance of such metrics with respect to process understandability. Finally, it is interesting to observe that the regression model contains metrics (e.g., Sz, $Scfc$, $Scnc$, Ls) measuring different process properties (e.g., process structure size / complexity / flow, conformance and terms). This indicates that there does not exist a unique process model property that predicts process understandability.

Table 3.11 shows the accuracy evaluation measures MAE and $Pred_5(1)$ computed by averaging the measures obtained in the different iterations of the leave-one-out cross-validation procedure, as well as those obtained by considering the Digistore's process models as $data_{test}$ set. Figure 3.19 plots the predicted and the actual values for all the predictions built by considering the whole set of metrics PrM_{all}. We observe that, on average, in case of PrM_{all} the error (MAE) done in the prediction is in the range [9-14]%. It means that, on average, the prediction is far from the actual value less than half of a point of the Likert scale, in case of the prediction evaluation based on Digistore, and about $\frac{3}{4}$ of the Likert scale point in case of the leave-one-out approach. Moreover, by

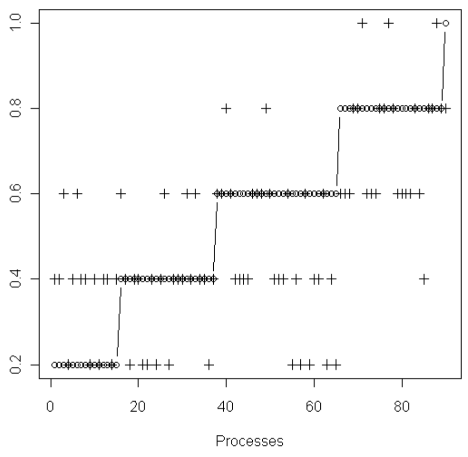


Figure 3.19: Plot of predicted (cross points) and actual (circle points) MO values for PrM_{all}.

	Leave-one-out		**Digistore-based**	
Model	MAE	$Pred_5(1)$	MAE	$Pred_5(1)$
PrM_{all}	0.14	0.91	0.09	0.93
PrM_{pca}	0.14	0.95	0.14	0.93

Table 3.11: Accuracy of the predictions

looking at the $Pred_5(1)$ value, we can notice that the percentage of predictions falling within the range ± 1 of the acutal modeller's opinion in the Likert scale is in the range [91-93]%, i.e., only less than 10% of predictions has an error greater than one point in the Likert scale. In case of PrM_{pca}, results are quite similar: MAE is 0.14 and $Pred_5(1)$ in the range [93-95]%. These results make the prediction models moderately effective with respect to the considered applications, since one point in the Likert scale can be, in some cases, not only a matter of small opinion variations.

We hence investigated the capability of the prediction model to correctly evaluate a process model as understandable or not. By taking out the neutral values of modellers' opinions about model understandability (i.e., $MO = 3$ in the Likert scale), we classified the remaining values as *non-understandable* if they are in the range [1-2] of the Likert scale and *understandable* if they are in the range [4-5] of the scale. Table 3.12 reports the values of the accuracy measures obtained by computing the percentage of correct predictions (*accuracy*) with respect to the two categories, understandable and non-understandable.

Model	**Leave-one-out** Accuracy	**Digistore-based** Accuracy
PrM_{all}	82.3	100
PrM_{pca}	91.9	88.9

Table 3.12: Accuracy of the predictions obtained by classifying modellers' opinions as understandable or non-understandable

Results show that, in case of the leave-one-out evaluation approach, predictions about understandability/ non-understandability opinions of process models are correct for the 82%, when using all the metrics for computing prediction, and for 92%, when using the PrM_{pca} metrics. In case of Digistore-based evaluation, results are strongly improved for PrM_{all} (full accuracy is reached), while for PrM_{pca} they show a trend similar to the one of the leave-one-out approach (88.9%). Hence, though the investigated metrics are moderately good predictors of the understandability degree of recovered process models, these last results suggest the suitability of their use as predictors of process model understandability/non-understandability, as well as encourage further investigations (e.g., with classification techniques as Support Vector Machines).

Overall, considering the obtained results, we can partially positively answer to question **RQ2**, though further investigations (e.g., by exploiting other prediction systems), are required, in order to understand if and to what extent the proposed metrics, properly combined, are good predictors for recovered process models. We plan, in our future works, to investigate other classification techniques and to validate the obtained results on a larger experiment involving a higher number of recovered processes and human modellers.

Follow-up

The results obtained with the analysis of the relationship between process metrics and understandability confirm the outcome of the evaluation of the reverse engineering approach proposed in Section 3.1 and, in particular, of the motivation leading to the application of clustering techniques for improving process readability. Table 3.13 reports the expert opinions and the most relevant metrics (i.e., those metrics that strongly correlate with the understandability according to the performed experiment) of a flat and a modularized version (in detail, we applied the "sequence-alternative-loop" clustering) of the process underlying each of the six Web applications considered. Results show that the cluster-based process modularization improves the overall understandability. In other terms, the

WA	Model	$MOAvg$	Sz	Sd	$Scfc$
Softslate	flat	1.67	40	0.037	44
Softslate	clust	3.33	36	0.042	37
Communicart	flat	1	43	0.037	43
Communicart	clust	1.67	39	0.04	35
Erol	flat	1.33	60	0.023	72
Erol	clust	2.33	57	0.025	66
Avactis	flat	1	152	0.008	168
Avactis	clust	1.67	151	0.0084	165
Interspire	flat	2.33	80	0.019	101
Interspire	clust	3	78	0.001	96
Winestore	flat	2.67	42	0.04	55
Winestore	clust	2.67	42	0.04	49

Table 3.13: Understandability and process metrics: flat versus modularized models

average of modellers' opinions ($MOAvg$) is improved by process model clustering. On the other hand, from Table 3.13, we can also observe that the structural metrics having a strong correlation with understandability detect the improvement, if any (e.g., in case of Winestore, the almost negligible variations in the structural metrics values correspond to no difference in the average modellers' opinions). This example, hence, confirms that the process modularization obtained by applying the proposed clustering-based technique in general improves the process model understandability [149] and that structural process metrics can be used as indicators for detecting such an improvement.

Threats to validity

A number of threats affect the validity of the results obtained in the performed experimental study.

External validity threats concern the generalization of results. The major threat to the external validity is due to the limited number of process models analysed in the experiment. A larger dataset of processes could have highlighted different connections between the process characteristics measured by metrics and the understandability. This threat is particularly strong for results related to label metrics, since all the process models related to the same application actually contain a subset of the label set recovered from the specific application (except for labels of clustered sub-processes). The obtained results, presenting only a low positive correlation (20%) between understandability and $L0d$, while reporting a lack of a strong relation between label metrics and understandability, could hence be due to the fact that only five sets of labels, one per application, have been evaluated. We believe that further iterations of the experiment, in which heterogeneity of process model label sets is also taken into account, can better corroborate the obtained results. Another relevant threat that affects the generalization of the outcome is the limited number of human modellers involved in the experimentation and their job position as researchers. Unfortunately, it is difficult to find professional modellers voluntarily collaborating in this type of investigations. However, we believe that the strong correlation we obtained among the modellers' opinions encourage us in upholding the experiment outcome. Finally, a third threat to external validity concerns the representativeness of the application domain with respect to "all possible domains"; in fact, we considered three e-commerce applications. Further experiments need to be conducted in applications belonging to different domains.

Internal validity threats concern external factors that affect a dependent variable. Two main threats to validity related to the modeller understandability evaluation can influence the results. First of all, the choice of using only a subjective evaluation of the actual understandability of process models. In order to limit this threat, we supported modellers in the (subjective) evaluation by asking them to answer some comprehension questions. Another threat is the choice of a 5-point scale for the evaluation; in fact, a more fine/coarse scale could be considered and different results could be obtained.

Construct validity threats concern the relationship between theory and observation. Two main threats to construct validity can affect the experiment results: (i) the learning effect of the modeller in the process model evaluation; each modeller, in fact, evaluated 5 processes related to the same application; and (ii) the process model layout. We tried to limit the first problem by using an experiment design based on five laboratories performed in different days and the second by manually adjusting the layout of process models used in the experiment.

Conclusion validity threats concern the relationship between treatment and outcome. A possible threat that may affect the drawn conclusions is related to the small sample size considered in the experiment (i.e., the limited number of processes and modellers), which may limit the capability of statistical test to reveal effects.

Chapter 4

Business Process Semantic Annotation

"Why is a raven like a writing desk?"
Lewis Carrol

Semantic Business Process Management (SBPM) [78, 56] aims at improving the level of automation in the specification, implementation, execution, and monitoring of business processes by extending business process management tools with the most significant results from the area of semantic web. The importance of semantic information in business process management is hence recognized starting from the design stage, i.e., the phase in which business processes are specified at an abstract (descriptive and non-executable) level. Thomas and Fellman [167] argued that annotating process descriptions with a set of tags taken from a set of domain ontologies would provide an additional support to the business analysis during the modelling activity. As remarked by Born et al [22], in fact, semantic annotation of business processes allows analysts to give a precise meaning to the process elements they are modelling, thus improving, among others: the reuse of parts of process models when creating new models; the detection of cross-process relations; the management of change; and providing a structured basis for knowledge transfer and for enabling automated reasoning on the process and its properties.

On the other hand, the semantic annotation of business process models is a time and resource consuming activity. It requires business designers to browse the source of the semantic knowledge (e.g., the domain ontology) and to select the appropriate annotation. Moreover, in some cases, the semantic knowledge to be used for the annotation is not available in a structured form, thus imposing an extra effort for its formalization. Automatic mechanisms (e.g., tools able to suggest candidate semantic annotations or to help in the construction or extension of structured knowledge sources) would hence be useful to support business designers and analysts in the semantic annotation activity.

In this chapter, we provide a description of our approach for the semantic enrichment of BPMN processes and for their formalization into a knowledge base with the aim of enabling automated reasoning on them and their properties (Section 4.1). Moreover, we propose a set of techniques for the automated suggestion of semantic annotations to business designers and for the enrichment of existing ontologies with missing concepts, when needed (Section 4.2).

The material related to the knowledge base formalization presented in this chapter has been published in [44, 45, 46], while the material related to the semantic suggestions in [50, 51].

4.1 Semantic Annotation of BPMN Process Models

Labelling of activities in BPMN is not a rigorous and well documented task: it is often performed with freedom [163] and subjectivity, thus generating unclear labels with mismatching, overlapping and often specific [115] (and hence difficult to understand outside the specific context) terms. The consequence is that human comprehension [115] of business processes, as well as acquisition of knowledge, become harder. Moreover, the information conveyed by labels lacks of structure and formality, thus resulting inaccessible to machines and preventing automated reasoning on processes and their properties.

In this section we describe our proposal to cope with this issue. In detail, we suggest to enrich BPMN business processes with domain annotations, thus clarifying the process domain semantics (Subsection 4.1.1), and to encode the annotated processes into an OWL knowledge base, thus providing a starting point for exploiting reasoning on the processes (Subsection 4.1.2).

4.1.1 Enriching BPMN Processes with Semantic Annotations

Business process models are mainly focused on the representation of activities, performers, as well as control and data flows. Domain information is conveyed only as informal labels. However, process element labelling is usually not rigorously performed by designers, thus resulting (e.g., in cases of large business processes) in situations of label inconsistency. It may happen, in fact, that tasks with different labels are used to represent the same activity or that different labels are used for describing different specializations of the same activity, adding irrelevant information with respect to the considered abstraction level.

Moreover, the amount of information that can be encoded in a human readable label is necessarily limited.

In order to deal with the problem of providing business process elements with domain knowledge that is both clear for humans and accessible to machines (thus enabling automated reasoning mechanisms), we propose to enrich process elements with annotations characterized by a semantics explicitly organized in a structured source of knowledge, i.e., with semantic concepts belonging to a (set of) domain ontology(es). Semantic annotations, in fact, can be used to provide a precise, formal meaning to process elements.

We graphically represent the semantic annotation of business processes and, in particular, of their underlying Business Process Diagrams (BPDs), by taking advantage of the BPMN *textual annotations*. In detail, we propose a semantic variant of BPMN, in which the ontology concept associated to a BPD element is prefixed by an "@" symbol. Such annotations allow us to categorize BPD elements, by unifying labels that represent the same concept and abstracting them into meaningful generalizations.

An example of a semantically annotated BPD is shown in Figure 4.1. It represents a process for realizing an assembled product starting from three raw products. Before the purchase data can be stored (sub-process "Store Purchase Data") and the task for product assembly ("Assemble Products") is executed, the three control flows for the acquisition of the raw products need to be executed in parallel and completed.

In this example, for instance, the tasks starting the three parallel flows for the raw product purchase, though exhibiting different labels ("Look For Product in the Warehouse", "Check Product Availability" and "Search for Product"), represent the same concept, i.e., all of them check whether the product is available in the warehouse. In this case, a semantic annotation, would allow to unify the semantics of the three tasks, though preserving their original labels. For example, assuming to have an ontology describing this domain and containing a to_check_product_availability concept, the three tasks could be semantically annotated by this, more general, concept. A similar argument can be used also for the other elements in the process.

4.1.2 Formalizing Semantically Annotated BPMN Processes

Semantic information is crucial for activities that involve reasoning and require automated support [78], as for example documenting or querying a process [49], enforcing a policy, or verifying constraints on the business logics [45]. In order to enable automated reasoning on a semantically annotated BPD and hence to automate the activities listed above, we encode the process into a logical knowledge base, the *Business Process Knowledge Base*

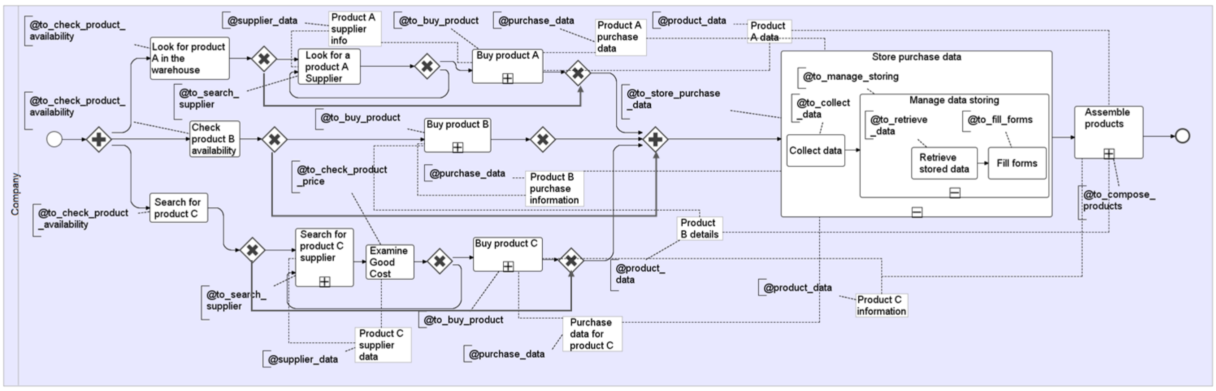


Figure 4.1: An example of a semantically annotated BPMN process

(BPKB).

A BPKB, schematized in Figure 4.2, is composed of four modules: a BPMN ontology, a domain ontology, a set of constraints and the BPD instances. We have implemented the BPKB using the standard semantic web language OWL (Web Ontology Language) based on Description Logics [12] (OWL-DL). Description Logics (DL) are a family of knowledge representation formalisms which can be used to represent the terminological and assertional knowledge of an application domain in a structured and formally well-understood way. The terminological knowledge, contained in the so-called Tbox, represents the background knowledge and the knowledge about the terminology (classes and properties) relevant for the described domain. The assertional part, the so-called Abox, contains knowledge about the individuals which populate the given domain in the form of membership statements. In our case, the terminological part (Tbox), which is the stable description of the domain, is provided by the upper level modules of Figure 4.2. Instead, the changeable part, which corresponds to a specific process description, is provided in the form of assertional knowledge (Abox).

The BPMN Ontology

The BPMN ontology, hereafter called BPMNO[1], formalizes the structure of a BPD. It is a formalization of the BPMN standard as described in Annex B of [131], and consists of a set of axioms that describe the BPMN elements and the way in which they can be combined for the construction of BPDs. The taxonomy of the graphical elements of BPMNO is illustrated in Figure 4.3. The ontology has currently the expressiveness of $\mathcal{ALCHOIN}(\mathcal{D})$ and a detailed description is contained in [66]. We remark that BPMNO

[1]Available for download at `http://dkm.fbk.eu/index.php/BPMN_Related_Resources`

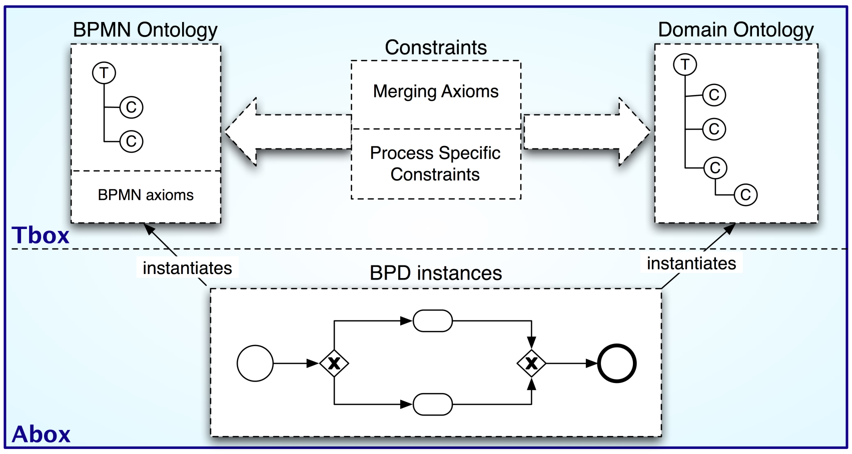


Figure 4.2: The Business Process Knowledge Base

provides a formalization of the structural part of BPDs, describing which are the basic elements of a BPD and how they are (can be) connected. BPMNO is not intended to model the dynamic behaviour of BPDs (that is, how the flow proceeds within a process). Ontology languages are not particularly suited to specify behavioural semantics. This part can be better modelled using formal languages for Workflow or Business Process Specification based on Petri Nets, as proposed in [93].

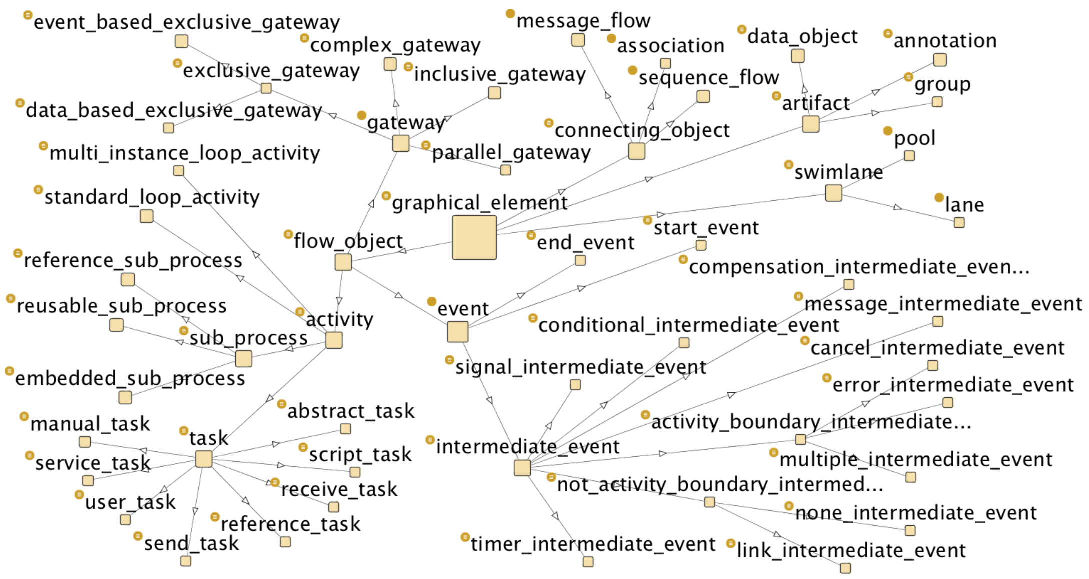


Figure 4.3: The graphical elements of BPMNO

The Domain Ontology

The domain ontology component, hereafter called BDO, consists of a (set of) OWL ontology(es) that describes a specific business domain. It allows to give a precise semantics to the terms used to annotate business processes. The BDO can be an already existing business domain ontology (e.g., RosettaNet or similar standard business ontologies), a customization of an existing ontology, or an artefact developed on purpose. Top level ontologies such as DOLCE [63] can be included as "standard" components of the domain ontology and used to provide typical annotation patterns to the BPD objects.

The Constraints

Constraints are used to ensure that important semantic structural requirements of process elements are satisfied. We distinguish between two different kinds of constraints: **merging axioms** and **process specific constraints**. Merging axioms state the correspondence between the BDO and the BPMNO. They formalize the criteria for correct/incorrect semantic annotations. Process specific constraints are expressions used to state specific structural requirements that apply to the process under construction. Differently from merging axioms, these expressions can have many different forms to match the specific properties of the process.

The BPD Instances

The BPD instances (or BPD objects) component of the BPKB consists of a set of ontology individuals and assertions which represent the elements of an annotated BPD in terms of instances of BPMNO and BDO classes. In order to clarify the description of this component we consider the (small) fragment of process reported in Figure 4.4. It represents the sub-process that manages the addition/removal of items in the shopping cart of the on-line shopping process. In detail, each graphical object g of a semantically annotated BPD β corresponds to an ontology individual in the set of BPD instances and to a set of assertions, both contained in A_β, the Abox which formalizes the BPD β (for example, the main part of the Abox associated with the sub-process in the example in Figure 4.4 is shown in Figure 4.5).

The assertions on the β's graphical objects can be divided into three groups: *BPM-type assertions*, *BPM-structural assertions* and *BPM-semantic assertions*. The first two groups of assertions involve concepts from BPMNO only, while the third group involves concepts from BDO only.

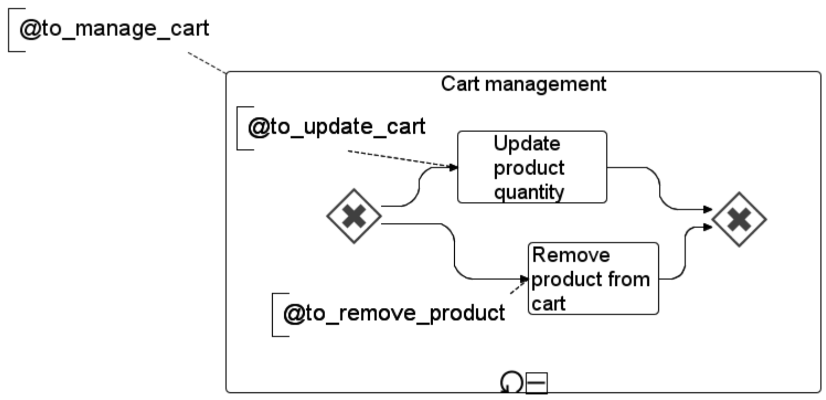


Figure 4.4: A sub-process for the cart management in an on-line purchase process

BPM-type assertions are used to store informations on the BPMNO-type[2] of a graphical object g. For every graphical element g of type T occurring in β, A_β contains the assertions $T(g)$, i.e., g is an instance of concept T[3]. For instance, we represent the fact that the gateway on the left in Figure 4.4 is an exclusive gateway with the BPM-type assertion data_based_exclusive_gateway(g_1) in Figure 4.5. Similarly, the assertion sequence_flow(s_1) states that the BPMNO-type of $s1$ is sequence_flow.

BPM-structural assertions are used to store information on how the graphical objects are connected. For every connecting object c of β that goes from a to b, A_β will contain two structural assertions of the form has_sequence_flow_source_ref(c, a) and has_sequence_flow_target_ref(c, b). For instance, the assertion has_sequence_flow_source_ref (s_2, g_1) in Figure 4.5, states that the sequence flow s_1 originates from the gateway g_1.

Finally, *BPM-semantic assertions* are used to store information on the BDO-type[4] of a BPD element, which is described by the semantic annotation associated with the BPD object. For every graphical element g of the diagram β which is annotated with a label C (where C is a BDO concept), A_β contains the assertion $C(g)$. For instance, the assertion to_update_cart(t_1) states that task t_1 in Figure 4.5 is an instance of the concept to_update_cart and is obtained from the semantic annotation to_update_cart of the sub-process in Figure 4.4.

[2]The term BPMNO-type specifies the BPMNO type of a BPD object, i.e., the BPMNO class/superclass of the corresponding instance in the BPKB.

[3]For the sake of readability, we omit the BPMNO prefix in non ambiguous expressions.

[4]Similarly to the term BPMNO-type, the term BDO-type specifies the BDO type of a BPD object, i.e., the BDO class/superclass of the corresponding instance in the BPKB.

BPD objects	
p_1 corresponds to the entire sub-process	
$s_1, \ldots, s_4$ correspond to the four sequence flows	
g_1 and g_2 correspond to the left and right gateways	
t_1 and t_2 correspond to the top and bottom atomic task	
BPM-type assertions	
embedded_loop_sub_process(p_1)	*/* p_1 is an iterative sub-process */*
data_based_exclusive_gateway(g_1)	*/* g_1 is a data-based XOR gateway */*
data_based_exclusive_gateway(g_2)	
sequence_flow(s_1)	*/* s_1 is a sequence flow object */*
sequence_flow(s_2)	
sequence_flow(s_3)	
sequence_flow(s_4)	
task(t_1)	*/* s_1 is an atomic task object */*
task(t_2)	
BPM-structural assertions	
has_embedded_sub_process_sub_graphical_elements(p_1, g_1)	
⋮	*/* p_1 contains g_1, g_2, $s_1 \ldots s_4$, t_1 and t_2 */*
has_embedded_sub_process_sub_graphical_elements(p_1, t_2)	
has_sequence_flow_source_ref(s_1, g_1)	
has_sequence_flow_target_ref(s_1, t_1)	
has_sequence_flow_source_ref(s_2, g_1)	
has_sequence_flow_target_ref(s_2, t_2)	
has_sequence_flow_source_ref(s_3, t_1)	
has_sequence_flow_target_ref(s_3, g_2)	
has_sequence_flow_source_ref(s_4, t_2)	
has_sequence_flow_target_ref(s_4, g_2)	
BPM-semantic assertions	
to_manage_cart(p_1)	*/* p_i is an activity of managing of carts */*
to_update_cart(t_1)	
to_remove_product(t_2)	

Figure 4.5: The encoding of the to_manage_cart sub-process in an OWL Abox

4.1.3 Automatically encoding a BPD into an Abox

We developed a tool for the automated transformation of a BPD into an OWL Abox. Given BPMNO, BDO and an annotated BPD β, the tool creates the Abox A_β and populates the ontology with instances of BPMN elements belonging to the specific process.

The input BPMN process is currently described in a `.bpmn` file, one of the files generated by both the Eclipse SOA Tools Platform and the Intalio Process Modeler tools. The `.bpmn` file is an XML file that contains just the structural description of the process, leaving out all the graphical details. The ontology is populated by parsing the file and instantiating the corresponding classes and properties in the BPKB Tbox.

The mapping between the XML elements/attributes used in the `.bpmn` file of the Eclipse tool and concepts and properties in the BPKB Tbox is realized by means of a mapping file. It associates each XML element of the `.bpmn` file to the corresponding concept in BPMNO and each of its attributes and child elements to the corresponding concept

```
<bpmn:Pool class="pool">
    <attributes>
        <this prop="has_BPMN_element_id" range="object"/>
        <this.object prop="this.object->has_object_id"/>
        <this prop="has_pool_process_ref" range="process"/>
        <this.process prop="this.process->has_process_name"/>
        <documentation prop="has_BPMN_element_documentation"/>
        <name prop="has_swimlane_name"/>
    </attributes>
    <relations>
        <lanes prop="has_pool_lanes" range="lane"/>
        <vertices prop="this.process->has_process_graphical_elements"
                  range="graphical_element"/>
    </relations>
</bpmn:Pool>
```

Figure 4.6: A fragment of the mapping file

or property, when this exists in the BPMNO. The fragment of mapping file in Figure 4.6 shows the correspondences between the pool process element (i.e., the XML element having type `bpmn:Pool` in the `.bpmn` file) and the BPMNO. Each XML element of this type in the `.bpmn` file will be translated into an instance of the concept BPMNO:Pool. The values of its attributes `name` and `documentation` will be the values of the two data properties has_swimlane_name and has_BPMN_element_documentation of the concept BPMNO:Pool. Moreover, the two values (instances of the classes BPMNO:Object and BPMNO:Process) of the BPMNO:Pool's object properties has_BPMN_element_id and has_pool_process_ref, will be instantiated by exploiting the unique id of the process element `pool`. Finally, the `pool`'s child elements annotated with the XML tags `lanes` and `vertices` will respectively be the values of the BPMNO:Pool's object property has_pool_lanes and of the BPMNO:Process's object property has_process_graphical_elements.

The BPMN process descriptions currently generated by the Eclipse or the Intalio tool do not exhaustively cover the features provided by the BPMN specification and, therefore, the full ontology potential. The mapping file is hence limited to the subset of the BPMN specification actually implemented by the considered tools and is based on assumptions implicitly made by the tools. Similarly, the mapping file depends on the particular process representation adopted by these tools and must be adjusted if a different tool is used for process editing.

Finally, the semantic annotations added to process elements and contained in the `.bpmn` file as XML elements of type `bpmn:TextAnnotation` are also used for populating the BDO concepts. By parsing the file, the process element associated to each XML element having type `bpmn:TextAnnotation` will be added as an instance of the BDO concept corresponding to the value of the semantic annotation, for each value prefixed by "@".

Our tool uses the `org.w3c.dom` XML parsing library to manage the `.bpmn` input file, Protégé[5] libraries to populate the resulting OWL Abox, and Pellet[6] for reasoning.

4.2 Semantic Annotation Suggestions

Despite the several potential advantages offered by the semantic enrichment of business processes, it presents a high cost. In fact, semantically annotating BPD elements, either at design time or after the process has been modelled, is resource and time consuming for business designers and analysts. It involves, for each BPD element to be annotated, browsing of the ontology and selection of the appropriate concept to be used for the element annotation. Moreover, it might happen that domain ontologies are not available or are incomplete, thus requiring designers to manually build or enrich them.

We use the linguistic analysis of the process element labels and of the concept names for providing semantic annotation suggestions to business designers. By taking advantage of these suggestions the experts are facilitated in adding semantic annotations to process elements without however changing their original and specific labels. This allows to limit the drawbacks deriving from the semantic annotation (e.g., time and resource consumption), while preserving its benefits.

After a short overview about concepts and notations used in the remainder of the section (Subsection 4.2.1), we describe our techniques for the semi-automated disambiguation of ontology concepts (three different algorithms for the analysis of the domain ontology aimed at extracting a WordNet "sense" for each of the concepts in the ontology are reported in Subsection 4.2.2), the annotation of process elements (an algorithm for the semi-automated semantic annotation of the process elements is reported in Subsection 4.2.3) and to support the business analyst in ontology creation and extension (Subsection 4.2.4). Finally, a preliminary evaluation of the proposed techniques is reported in Subsection 4.2.5.

[5]`http://protege.stanford.edu/`
[6]`http://clarkparsia.com/pellet/`

4.2.1 Background

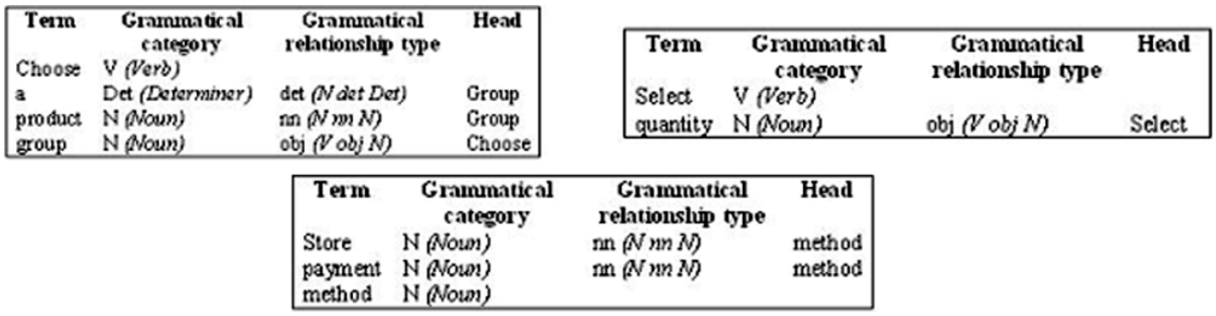
Linguistic Analysis

Natural language processing is a wide research area, including a number of different tasks and approaches. The analysis of short sentences, like those characterizing labels or ontology concepts, is one such task. Linguistic analysers, as for example MINIPAR[7], do not only allow to tokenize (short) sentences, reduce words to their stems and classify terms into grammatical categories (e.g., verbs, nouns, adjectives), but they are also able to find dependencies and grammatical relationships between them (e.g. verb-object, article-noun, specifier-specified). In detail, given a sentence s, MINIPAR is able to tokenize it, thus extracting its word list, $WS(s) = \{w_i \in Dict | s = w_1...w_n\}$, where $Dict$ is a given dictionary of words[8]. Moreover, for each word w_i, it identifies the grammatical category $mGCat(w_i) \in MGCS$, as well as the dependency relationship ($\rightarrow_R$, with $R \in MGREL$) with its *head word* ($head_w(w_i)$), if any. The head word of a word w_i, in fact, is the word in the sentence dominating the dependency relationship $\rightarrow_R$ with w_i, so that, if $w_i \rightarrow_R w_j$, then $w_j = head_w(w_i)$. The only word free from dependencies in a sentence s, is the *sentence head word* ($head_s(s)$). It represents the root of the dependency tree, a tree that, built upon the dependency relationships, connects all the terms in the sentence. $MGCS = \{V, N, V\_BE, A, ...\}$ is the set of the MINIPAR classification of grammatical categories (e.g., V = verb, N = noun, V_BE = "to be" verb, A =adjective or adverb, ...), while $MGREL$ is the set of the MINIPAR dependency relationships. $MGREL = MGREL_{term} \cup MGREL_{clause}$ is the union of the set of the dependencies between pairs of terms, $MGREL_{term} = \{subj, obj, nn, det, ...\}$, e.g., $subj$ = verb-subject (subject $\rightarrow_{subj}$verb), obj = verb-object (object$\rightarrow_{obj}$verb), nn = specified-specifier (specifier $\rightarrow_{nn}$ specified) or det = determined-determiner (*determiner* $\rightarrow_{det}$ *determined*) relationship, and the set of the relationships involving clauses, $MGREL_{clause} = \{rel, fc, ...\}$, e.g., rel = noun-relative clause (noun $\rightarrow_{rel}$ relative clause), verb-final clause (verb $\rightarrow_{fc}$ final clause) relationship.

Henceforth, we will refer to a verb of a parsed sentence with the character v (i.e., $MGCat(v) = V$) and to a noun with the character n (i.e., $MGCat(n) = N$). Moreover, we introduce the function $o(v)$ to denote the object of the verb v (i.e., $o(v) \rightarrow_{obj} v$) and $s(n)$ to represent the specifier of n (i.e., $s(n) \rightarrow_{nn} n$). For example, by applying MINIPAR to the short sentence "Choose a product group", we obtain the information in Figure 4.7 (top left); by applying it to the sentence "Select quantity", we get the results shown in

[7]`http://www.cs.ualberta.ca/~lindek/minipar.htm`.
[8]MINIPAR takes advantage of WordNet.

Figure 4.7 (top right).



Term	Grammatical category	Grammatical relationship type	Head
Choose	V *(Verb)*		
a	Det *(Determiner)*	det *(N det Det)*	Group
product	N *(Noun)*	nn *(N nn N)*	Group
group	N *(Noun)*	obj *(V obj N)*	Choose

Term	Grammatical category	Grammatical relationship type	Head
Select	V *(Verb)*		
quantity	N *(Noun)*	obj *(V obj N)*	Select

Term	Grammatical category	Grammatical relationship type	Head
Store	N *(Noun)*	nn *(N nn N)*	method
payment	N *(Noun)*	nn *(N nn N)*	method
method	N *(Noun)*		

Figure 4.7: Information extracted by MINIPAR

Unfortunately, short sentences are intrinsically difficult to analyse through linguistic processing, because they carry limited and compact information. Sometimes, it happens that the analysis performed by the parser is wrong or inaccurate. For example, parsing of the label "Store payment method" by means of MINIPAR gives the (partially wrong) result shown in Figure 4.7 (bottom).

WordNet

The simple parsing of a sentence executed by MINIPAR is usually not enough for determining its semantics. The same term, in fact, can have multiple meanings (*polisemy*), as well as more terms (*synonyms*) can represent the same concept. WordNet [118] is one of the most known resources allowing to categorize terms according to their meaning (sense) and synonym set (synset). The same word, in fact, can be used as the representative for different Parts Of the Speech (*POS*) and for each of them, it can have multiple senses. The information gathered from *POS* type and sense allows to disambiguate the word meaning. A *WordNet Dictionary*, *WNDict*, is therefore a collection of (word, word-sense, word-type) triples ($WNDict = \{(w_i, s_j, wnpos) | w_i \in Dict \wedge wnpos \in WNPOS\}$, where $WNPOS = \{N, V, Adj, Adv\}$ is the WordNet set of different type categories, e.g., N = noun, V= verb, Adj = adjective, Adv = adverb). Each triple $(w_i, s_j, wnpos)$ identifies a unique synset including $(w_i, s_j, wnpos)$, as well as all the other (if any) WordNet triples $(w_u, s_v, wnpos)$ having the same, specific meaning of $(w_i, s_j, wnpos)$. Hereafter we will use the function senses ($senses(w_i, wnpos) = \{s_j | (w_i, s_j, wnpos) \in WNDict\}$) to compute all the senses of a word w_i belonging to a specific type category $wnpos$, the function $wnPOS : MGCS \rightarrow WNPOS$ to map a subset of the MINIPAR categories $MGCS' \subseteq MGCS$ to the WordNet type categories (the main mappings of the function

$WNPOS/MGCS'$	V	N	V_BE	A
V	$*$		$*$	
N		$*$		
Adj				$*$
Adv				$*$

Table 4.1: Mapping from $MGCS'$ to $WNPOS$: function $wnPOS$.

are summarized in Table 4.1) and the function $SynsetRepr(syn) = (w_i, s_j, wnpos)$, where syn is a synset, to denote the synset canonical representative.

Information Content Similarity Measure

The information content similarity approach is based on the term information content: the more frequently a term occurs, the less information it conveys. The information content can be measured as the negative logarithm of the normalized term frequency. Given two concepts, their semantic similarity depends on the amount of information they share. Assuming we can map them onto a hierarchical structure, such as WordNet, the semantic similarity is given by the information content of the Most Specific Common Abstraction ($MSCA$). The information content of a term can be measured on the basis of the term occurrences in large text corpora (normalized with respect to the hierarchical structure). An approximation of such a measure can be obtained by analysing the hierarchical structure and counting the number of hyponyms, under the assumption that a term with lots of hyponyms tends to occur quite frequently in large corpora. In this work, we approximate the probability of terms by using hyponyms in WordNet: $p(t) = \frac{hypo(t)+1}{max_{WN}}$, where max_{WN} is the number of hyponyms of the term t and max_{WN} is the total number of WordNet words. One of the most used ways of computing the information content similarity between two terms t_1 and t_2 is Lin's formula [101]:

$$ics(t_1, t_2) = \frac{2 * \log(p(MCSA(t_1, t_2)))}{\log(p(t_1)) + \log(p(t_2))}$$

Henceforth, when talking about the information content similarity, we refer to Lin's formula.

4.2.2 Domain Ontology Analysis

In the literature, several *WSD* (Word Sense Disambiguation) algorithms have been proposed. They determine senses for words appearing in large corpora of texts, written in natural language [83]. Only a few WSD works, in particular in the context of the semantic web [187], deal with the mapping between ontology concepts and WordNet synsets. In order to simplify the semantic annotation of process activities and make it more accurate, we also need to solve the semantic ambiguity of ontology concepts, by mapping them to unique synsets. For this purpose, we exploit the information we can gather from the BDO itself (in particular from its hierarchical structure) and compare it with the WordNet taxonomy. However, the mapping, as well as the comparison, presents several issues. The first challenge is the structural difference between strings representing ontology concepts and words with a specific sense characterizing a synset. A concept name can be a single word or a short sentence describing the concept. While in the first case the concept name is mapped to one of the synonyms in a synset (the sense with the highest similarity value), in the second case, the short sentence needs to be analysed linguistically in order to mine the word that, representing the dominant meaning of the concept, also determines its ontology relationships (in particular its is_a relationship).

Let us consider, for example, the BDO concept to_record_information. By assuming that concept names have been meaningfully assigned, the concept will very likely represent the action of storing information. Therefore, the concept head word is the verb "to record", which probably has an is_a relationship with some action or event concept in the ontology (see for example SUMO[9] or OntoSem[10]).

The word representing the dominant meaning in a short sentence can be mined by applying a linguistic analyser, as for example MINIPAR, to the sentence itself. It will be the root word in the dependency tree produced by the parser, i.e., the sentence head word.

Once the head word has been identified for each concept, the concept has to be mapped to a WordNet synset. To this purpose we propose three approaches: two of them take advantage of the ontology hierarchical structure, while the third requires the existence of comments and/or labels in the BDO. Due to their different kind, the first two approaches can be complemented by the third and vice-versa.

We do not expect that automated disambiguation of ontology concepts is completely error free, especially because we deal with short sentences, while the available approaches have been proven to work well with long texts in natural language. Moreover, it only allows disambiguating the sentence head word of the concept (i.e., if the concept name is a sentence only the sentence head word is considered for the disambiguation). Hence, the automatically produced disambiguation may need to be revised by the user before moving to the next step of process annotation.

[9]http://protege.stanford.edu/ontologies/sumoOntology/sumo_ontology.html
[10]http://morpheus.cs.umbc.edu/aks1/ontosem.owl

Relative Disambiguation Algorithm

In order to determine the sense of the sentence head word of each ontology concept, the *Relative Disambiguation Algorithm* (RDA) exploits the information coming from each of the concepts having a relationship (i.e., parent-child, sibling-sibling or child-parent) with the current node in the hierarchical structure.

In detail, for each concept c we consider its similarity with concepts of the same synset type and belonging to its relative concept set $RC(c) = PC(c) \cup SC(c) \cup CC(c)$, where $PC(c)$ is the set of the super-concepts of c, $SC(c)$ the set of sibling concepts of c and $CC(c)$ the set of sub-concepts of c. Given the synset type of the sentence head word $\overline{w}_c$ of the current concept c (it can be inferred from the grammatical category of w_c), for each sense s_i of such a word and for each relative concept $rc \in RC(c)$ of the same type, we compute the maximum information content similarity value $maxics((\overline{w}_c, s_i), \overline{w}_{rc})$ (see Subsection 4.2.1) between the two head words ($\overline{w}_c$ and $\overline{w}_{rc}$) with respect to all the possible senses of $\overline{w}_{rc}$. The identified synset, characterized by the pair ($\overline{w}_c$, s_i) chosen for the current concept c, will be the one with the highest average of $maxics((\overline{w}_c, s_i), \overline{w}_{rc})$ computed over all the relative concepts of c. Algorithm 1 reports the pseudo-code of the proposed algorithm.

Algorithm 1 Relative Disambiguation Algorithm (RDA)

```
Input $DO$: Domain Ontology
Input $WNO$: WordNet Ontology
Input $c$: ontology concept
Output $s^*$: sense for the head word of the concept $c$
 1: $\overline{w}_c = head_s(c)$
 2: $\overline{w}_c\_type = wnPOS(mGCat(\overline{w}_c))$
 3: for each $s_i \in senses(\overline{w}_c, \overline{w}_c\_type)$ do
 4:     $RC(c) = PC(c) \cup SC(c) \cup CC(c)$
 5:     $rc\_n = 0$
 6:     for each $rc \in RC(c)$ do
 7:         $\overline{w}_{rc} = head_s(rc)$
 8:         $\overline{w}_{rc}\_type = wnPOS(mGCat(\overline{w}_{rc}))$
 9:         if $\overline{w}_{rc}\_type = \overline{w}_c\_type$ then
10:             $maxics((\overline{w}_c, s_i), \overline{w}_{rc}) = max_{s_j \in senses(\overline{w}_{rc}, \overline{w}_{rc}\_type)} ics_{Lin}((\overline{w}_c, s_i), (\overline{w}_{rc}, s_j))$
11:             $rc\_n = rc\_n + 1$
12:         end if
13:     end for
14:     $avgics(\overline{w}_c, s_i) = \frac{\sum_{rc \in RC(c)} maxics((\overline{w}_c, s_i), \overline{w}_{rc})}{rc\_n}$
15: end for
16: $s^* = arg_{max}(max_{s_i \in senses(\overline{w}_c, \overline{w}_c\_type)} avgics(\overline{w}_c, s_i))$
17: return $s^*$
```

Let us consider the ontology on the left in Figure 4.8 and let us assume we are interested in computing the WordNet sense associated with the concept c_4. The sentence head word of the concept (the root of the parsing tree) is, for example, the verb v_4. We also assume that the sentence head words of its parent c_1 (v_1), its sibling concept c_2 (v_2) and its child

$\overline{w}_c$	$\overline{w}_c$'s senses	$\overline{w}_{rc}$	$\overline{w}_{rc}$'s senses	$ics((\overline{w}_c, s_i), \overline{w}_{rc})$	$maxics$	$avgics$
v_4	#1	v_1	#1	0.2	0.8	0.7
			#2	0.4		
			#3	0.8		
		v_2	#1	0.5	0.5	
		v_5	#1	0.7	0.7	
			#2	0.3		
		v_6	#1	0.2	0.8	
			#2	0.8		
	#2	v_1	#1	0.4	0.4	0.65
			#2	0.1		
			#3	0.0		
		v_2	#1	0.5	0.5	
		v_5	#1	0.8	0.8	
			#2	0.4		
		v_6	#1	0.9	0.9	
			#2	0.3		

Table 4.2: Example values for the RDA

concepts c_5 and c_6 (v_5 and v_6, respectively) are all verbs. Table 4.2 shows some possible similarity values between each sense of v_4 (#1 and #2) and each sense of the sentence head words of each of its relative concepts. For each relative concept, the maximum similarity measure over all its senses has been computed (sixth column in the table). Finally, in the seventh column, the average of these similarities has been computed for each sense of v_4. Since the maximum value is 0.7, the sense chosen for the verb v_4 (in the concept name c_4) is #1.

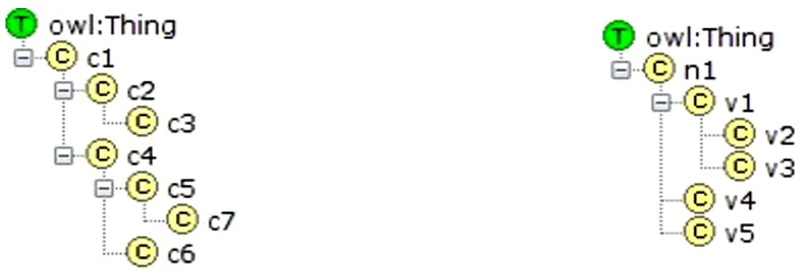


Figure 4.8: Examples of ontologies

Open Problems

Although it may work well in practice, the algorithm presented above has a limitation: the sense that it infers for each concept of the ontology (on the basis of the concept relationships with other ontology concepts) is independent from the senses inferred for the other concepts. Though the sense of a single concept is computed on the basis of the senses of its relative concepts, the final sense assigned to each of the relative concepts could be different from the one used for inferring the sense of the current concept. For example, let us consider again the ontology in Figure 4.8 (on the left) and the concept c_6, whose

$\overline{w}_c$	$\overline{w}_c$'s senses	$\overline{w}_{rc}$	$\overline{w}_{rc}$'s senses	$ics((\overline{w}_c, s_i), \overline{w}_{rc})$	$maxics$	$avgics$
v_6	#1	v_4	#1	0.2	0.9	0.9
			#2	0.9		
		v_5	#1	0.9	0.9	
			#2	0.6		
	#2	v_4	#1	0.8	0.8	0.55
			#2	0.3		
		v_5	#1	0.3	0.3	
			#2	0.1		

Table 4.3: An example of possible problems raised by RDA

sentence head word is v_6. Table 4.3 shows the information content similarity between each of the senses of v_6 and each of the senses of its relatives (c_1 and c_2). Moreover, it reports the highest similarity with respect to the relative senses, and the average over the relatives for each sense of v_6. The algorithm will choose the sense of v_6 with the maximum *avgics*, i.e., #1. However, by looking at Table 4.2, we can notice that the choice of the sense #1 for the concept v_4 was based on its maximum similarity with $(v_6, \#2)$: $ics((v_4, \#1), (v_6, \#2)) = 0.8$. On the contrary, by fixing the sense of v_6 (sense #1), the maximum *avgics* of v_4 would have been the one related to the sense #2 ($ics((v_4, \#2), (v_6, \#1)) = 0.9$, while $ics((v_4, \#1), (v6, \#1)) = 0.2$).

In order to deal with this problem, we propose a second algorithm. Due to the excessive computational effort deriving from the exploration of the whole set of possible combination of senses among the sentence head words of all the concepts in the ontology, it limits the information used for the sentence head word disambiguation to the child nodes of the current concept.

Child Disambiguation Algorithm

The *Child Disambiguation Algorithm* (CDA) computes the sense of (the sentence head word of) a concept on the basis of similarities with concepts having a child-parent relationship with the current concept.

It is based on a bottom-up visit of the ontology graph, from leaves up to the root. The first senses to be computed are those related to leaf concepts. To perform this computation, we consider each of the possible senses of the leaf's parent. For each parent sense, we maximize the semantic similarity with the senses of each child node and take the sum over the children. The parent sense with maximum sum is chosen for this node and determines the children's senses.

For intermediate nodes, the node's sense can be computed as the sense maximizing the sum of the semantic similarities with the synsets associated to the sentence head word of each of its children. Algorithm 2 and 3 represent the pseudo-code description of the algorithm, based on the recursive visit function visit_concept.

Algorithm 2 visit_concept(c)

```
Input $DO$: Domain Ontology
Input $WNO$: WordNet Ontology
Input $c$: ontology concept
Output $s^*$: sense for the head word of the concept $c$
1: if $CC(c) = \emptyset$ then
2:     return null
3: end if
4: for each $cc \in CC(c)$ do
5:     $visit\_concept(cc)$
6: end for
7: $\overline{w}_c = head_s(c)$
8: $\overline{w}_c\_type = wnPOS(mGCat(\overline{w}_c))$
9: $s^* = arg_{max}(max_{s_i \in senses(\overline{w}_c, \overline{w}_c\_type)}$
                    $\sum_{cc \in RC(c) | wnPOS(mGCat(head_s(cc))) = \overline{w}_c\_type}$
                        $ics_{Lin}((\overline{w}_c, s_i), (head_s cc, comp\_sense(cc, (\overline{w}_c, s_i)))))$
10: $set\_sense(c, s*)$
11: for each $cc \in CC(c)$ do
12:     if $is\_leaf(c)$ then
13:         $set\_sense(cc, get\_sense(head_s(cc), (\overline{w}_c, s^*)))$
14:     end if
15: end for
16: return $s^*$
```

Algorithm 3 comp_sense(c,($\overline{w}$,s))

```
Input $WNO$: WordNet Ontology
Input $c$: ontology concept
Input $(\overline{w}, s)$: pair head word, sense for the parent concept of $c$
Output $s^*$: sense for the head word of the concept $c$
1: if $is\_leaf(c)$ then
2:     $\overline{w}_c = head_s(c)$
3:     $\overline{w}_c\_type = wnPOS(mGCat(\overline{w}_c))$
4:     $s^* = arg_{max}(max_{s_j \in senses(\overline{w}_c, \overline{w}_c\_type)} ics_{Lin}((\overline{w}, s), (\overline{w}_c, s_j)))$
5:     return $s^*$
6: else
7:     return $get\_sense(c)$
8: end if
```

As in the previous example, let us consider the ontology on the left in Figure 4.8 and let us assume we are interested in computing the WordNet sense associated with the concept c_4, whose sentence head word (the root of the parsing tree) is v_4. Similarly to the previous example, let us also assume that the sentence head words of its descendants c_5, c_6 and c_7 are all verbs: v_5, v_6 and v_7, respectively. By applying the CDA, only child concepts (c_5 and c_6), have to be considered. One of them (c_6) is a leaf of the ontology graph, therefore the sense associated to its head word v_6, will be the one maximizing the current concept global similarity over all the children; the other concept, c_5, is not a leaf, therefore we assume a sense (#2) has already been assigned to its head word v_5 on the base of c_5's children (i.e., c_7). Table 4.4 shows some values related to this scenario. The sense for

$\overline{w}_c$	$\overline{w}_c$'s senses	$\overline{w}_{cc}$	$\overline{w}_{cc}$'s senses	$ics((\overline{w}_c, s_i), \overline{w}_{cc})$	$maxics$	$icssum$
v_4	#1	v_5	#2	0.3	0.3	1.1
		v_6	#1	0.2	0.8	
			#2	0.8		
	#2	v_5	#2	0.4	0.4	1.3
		v_6	#1	0.9	0.9	
			#2	0.3		

Table 4.4: Example values for the CDA

v_4 (#2) is computed as the one corresponding to the highest $icssum$ value (1.3), where $icssum$ is the sum, for each concept sense, of the $maxics$ values with its child concepts. Once the parent sense has been set, senses for child leaf concepts can also be assigned: in this case #1 is assigned to v_6 due to the highest information content similarity with $(v_4, \#2)$.

Open Problems

When a concept, whose sentence head word has to be mapped to a WordNet synset, does not have children in the ontology (i.e., is a leaf) and does not have a parent belonging to the same grammatical category, the algorithm fails in finding the mapping synset. In fact, in WordNet the hierarchy related to a synset of a given grammatical category belongs to the same category. Hence, if parent and child have different types, they also belong to different hierarchies, thus resulting in a non-existent WordNet is_a relationship between the domain ontology parent and child concept.

Let us consider the ontology in Figure 4.8 (right), where a concept name starting with "n" is used for denoting a sentence head word of the concept belonging to the noun grammatical category and a concept name starting with "v" is used for denoting a sentence head word of the concept belonging to the verb grammatical category.

Let us consider the leaves v_4 and v_5, belonging to the verb grammatical category and their parent n_1, belonging to the noun category. The corresponding synsets in WordNet belong to completely unrelated hierarchies (of verbs and nouns, respectively). Hence it is impossible to compute the similarity between parent and children in this case. This is also reflected in the Algorithm 2, which restricts the computation of the similarity to synsets of the same type.

This situation suggests the need to be able to exploit also other types of information in the ontology disambiguation phase.

Term	**Concept definition**	**Senses**	**Sense Description**	$Sim(c, (\overline{w}_c, s))$
search	to actively search for or seek	#2	search or seek	2/3 = 0.67
		#4	subject to a search	1/4 = 0.25
		#1	try to locate or discover, or try to estabilish the existence of	0.0

Table 4.5: Similarity computed on the basis of the shared terms between the search concept definition and the WordNet synset definition for each sense of the verb search when applying the DBDA.

Description-Based Algorithm

Often the domain ontology comes with concept descriptions, similar to those that are found in WordNet. This information can be exploited for the mapping between the two ontologies. We propose an algorithm, the *Description-Based Disambiguation Algorithm* (DBDA) that measures the similarity between the sentence head word of a concept in the ontology and a synset in the WordNet ontology on the basis of the number of terms shared by the two descriptions (the domain ontology and the WordNet one). In detail, once both descriptions have been tokenized and stemmed, the similarity between a concept and a WordNet synset is computed as the ratio between the cardinality of the intersection and the union sets of the two groups of terms produced by the linguistic analysis:

$$sim(c, (\overline{w}_c, s^*)) = \frac{|description(c) \cap description(\overline{w}_c, s^*)|}{|description(c) \cup description(\overline{w}_c, s^*)|}$$

The WordNet synset with the description most similar to the ontology concept determines the sense for that concept.

Table 4.5 shows the similarity measure, as defined above, between the description of the concept search in the OntoSem ontology and the descriptions characterizing each of the synsets associated to the senses of search in WordNet. The chosen sense suggested for the word is the one associated with the highest similarity, i.e., sense #2.

This algorithm can also be combined with either of the previous ones and used in a complementary way in order to deal with their limitations. The DBDA, for example, could be combined with the CDA disambiguation algorithm in cases in which a sense cannot be determined for nodes having only relationships with nodes of different grammatical categories. In general, both with RDA and CDA, when the similarity value is not significant enough (i.e., when it does not reach a given threshold), we can resort to DBDA disambiguation.

4.2.3 Business Process Semantic Annotation Suggestions

Once the ontology concepts are linked to a single WordNet sense, the business designer can be supported in the semantic annotation, by receiving suggestions for each process activity she intends to annotate. The choice of the suggestions is based on the semantic similarity between BPMN element label and ontology concepts: the higher the similarity measure, the higher the score given to the candidate ontology concept c as a possible semantic annotation for the element label l. Simpler criteria (e.g., those based on string or syntactic similarity), in fact, do not allow to capture more complex and meaningful mappings between pairs of natural language sentences, taking into account the semantics of the terms in the sentences.

The semantic similarity of a pair (l, c) can be based on the semantic similarity between pairs of words respectively in l ($W_l = \{w_i \in Dict | l = w_1 ... w_n\}$) and in c ($W_c = \{w_j \in Dict | c = w_1 ... w_m\}$). We define the *candidate set of pairs* CSP as $CSP \subseteq W_l \times W_c$ such that:

1. Each word $w_i \in W_l$ and $w_j \in W_c$ appears at most once in CSP;
2. $\forall\ (w_i, w_j) \in CSP$
   (a) w_i and w_j do not depend on any other term in their respective sentences;
   (b) if $w_i \to_R w_k$ and $w_j \to_R w_l$, then $(w_k, w_l) \in CSP$, $R \in MGREL$;
3. The total semantic similarity (i.e., the weighted sum of similarity values over each pair in CSP) is maximized by CSP.

We take advantage of the linguistic information available from linguistic analysis (parsing and synset computation) to choose proper candidate pairs (e.g., verbs are never paired with sentence objects, as well as objects are never paired with object specifiers), but also to give weights to the semantic similarity measures. In detail, the Business Process Modelling guideline, suggesting the use of the verb-object form for labelling activities (e.g., [117], [160], [105]), is the premise for our choice of limiting the clause analysis in the semantic annotation of an activity to verbs, objects and object specifiers and for ranking the three components according to their role (i.e., the verb has greater importance than the object, in turn more important than the specifier). Similarly, in the infrequent case of short sentences including also dependent clauses (e.g., "sign the order to send"), a weight, in inverse proportion with the dependency level, is assigned to clauses.

The enforcement of the constraints defined above and the different weights assigned to the different parts of the sentences, suggested us a simple heuristics to compute the

CSP with maximum match. We start from the roots of the two parsing trees, and in particular from the verbs, since these have higher weight. Specifically, we consider the verbs that are located higher in the hierarchy and we go down in the two trees following the grammatical dependencies.

Hence, the matching $CSP(W_l, W_c)$ between the words in label l and the words in concept c is built in the following steps:

1. given $V(l)$ and $V(c)$, the set of verbs in l and c respectively, the pair of verbs (v_l, v_c), with $v_l \in V(l)$ and $v_c \in V(c)$ located higher in the parsing trees of the sentences in l and c, is added to $CSP(W_l, W_c)$;

2. the pair composed of the respective objects[11] $(o(v_l), o(v_c))$, where $o(v_l) \rightarrow_{obj} v_l$ and $o(v_c) \rightarrow_{obj} v_c$ is added to $CSP(W_l, W_c)$ (when both verb objects exist);

3. the pair composed of the object specifiers $(s(o(v_l)), s(o(v_c)))$, where $s(o(v_l)) \rightarrow_{nn} o(v_l)$ and $s(o(v_c)) \rightarrow_{nn} (o(v_c))$ is added to $CSP(W_l, W_c)$ (when both object specifiers exist);

4. recursively, if the specifiers of the two sentences are, in turn, specified by another pair of specifiers $(s^*(o(v_l))$ and $s^*(o(v_c)))$, the pair $(s^*(o(v_l))$ and $s^*(o(v_c)))$ is added to $CSP(W_l, W_c)$;

5. steps 1-4 are possibly iterated for each pair of clauses $(cl_i, cl_j) \in CLS(l) \times CLS(c)$, respectively dependent on l and c via the same grammar relationship (e.g., cl_i and cl_j are both relative clauses of l and c, respectively) and maximizing their semantic similarity, where $CLS(l)$ and $CLS(c)$ are the sets of clauses dependent on l and c, respectively.

Algorithm 4 and 5 show the pseudo-code of the algorithm used for computing the CSP and the semantic similarity between the two sentences.

In practical cases, for short sentences such as those used in process labels and ontology concept names, there is typically at most one clause with one verb and its object ([22], [160], [105]). Hence, step (1) produces an initial CSP with at most one pair, containing the two verbs. Step (2) adds the pair of objects for the two verbs, when such objects exist in both short sentences. Finally, when in some cases, the object specifiers (at most one per sentence) also exist in the label and the concept name, the object specifier pair is added.

[11] We assume that every clause has at most one object and that the parser, in case of ditransitive verbs, is able to retrieve the “main” object.

Algorithm 4 analyze_sentences

```
Input $l$: activity label
Input $c$: ontology concept
Input $n$: clause nesting level
Output $CSP(W_l, W_c)$: candidate set pair between words in $l$ and $c$
Output $SemSim(l, c)$: semantic similarity between $l$ and $c$
 1: $cl_l = main\_clause(l)$
 2: $cl_c = main\_clause(c)$
 3: $\langle CSP, SemSim \rangle = analyze\_clause(cl_l, cl_c)$
 4: $CSP(W_l, W_c) \leftarrow CSP$
 5: $SS = SemSim$
 6: $n = n + 1$
 7: for each $R \in MGREL\_clauses$ do
 8:     $\left\langle CSP^R, SemSim^R \right\rangle = arg_{max}( \max_{\left\langle CSP^R_{i,j}, SemSim^R_{i,j} \right\rangle | \left\langle CSP^R_{i,j}, SemSim^R_{i,j} \right\rangle = analyze\_sentences(cl_i, cl_j) \wedge cl_i \in CLS(cl_l) \wedge cl_i \rightarrow_R cl_l \wedge cl_j \in CLS(cl_c) \wedge cl_j \rightarrow_R cl_c} SemSim^R_{i,j})$
 9:     $CSP(W_l, W_c) \leftarrow CSP^R$
10:     $SS = SS + DEP\_CLAUSE\_WEIGHT * SemSim^R$
11: end for
12: return $CSP(W_l, W_c)$, $\frac{SS}{1+n/DEP\_CLAUSE\_WEIGHT}$
```

Algorithm 5 analyze_clauses

```
Input $cl_l$: label clause
Input $cl_c$ :: concept clause
Output $CSP(W_{cll}, W_{clc})$: candidate set pair between words in $cl_l$ and $cl_c$
Output $SemSim(cl_l, cl_c)$: semantic similarity between $cl_l$ and $cl_c$
 1: $SS = 0$
 2: $n = 0$
 3: if $V(cl_l) \neq \emptyset$ then
 4:     $n = VERB\_WEIGHT$
 5:     if $V(cl_c) \neq \emptyset$ then
 6:         $v_{cll} \in V(cl_l) | v_{cll}$ is higher in the parsing tree
 7:         $v_{clc} \in V(cl_c) | v_{clc}$ is higher in the parsing tree
 8:         $CSP(W_{cll}, W_{clc}) \leftarrow (v_{cll}, v_{clc})$
 9:         $SS = VERB\_WEIGHT * ics_{Lin}(v_{cll}, v_{clc})$
10:         if $o(v_{cll})$ exists then
11:             $n = n + OBJECT\_WEIGHT$
12:             if $o(v_{clc})$ exists then
13:                 $SS = SS + OBJECT\_WEIGHT * ics_{Lin}(o(v_{cll}), o(v_{clc}))$
14:                 $n_{cll} = o(v_{cll})$
15:                 $n_{clc} = o(v_{clc})$
16:                 $i = 1$
17:                 while $s(n_{cll})$ exists do
18:                     $n = n + i * OBJ\_SPEC\_WEIGHT$
19:                     if $s(n_{cll})$ exists then
20:                         $CSP(W_{cll}, W_{clc}) \leftarrow (s(n_{cll}), s(n_{clc}))$
21:                         $SS = SS + i * OBJ\_SPEC\_WEIGHT * ics_{Lin}(s(n_{cll}), s(n_{clc}))$
22:                         $n_{cll} = s(n_{cll})$
23:                         $n_{clc} = s(n_{clc})$
24:                         $i = i/2$
25:                     end if
26:                 end while
27:             end if
28:         end if
29:     end if
30: end if
31: return $\langle CSP(W_{cll}, W_{clc}), \frac{SS}{n} \rangle$
```

$term_1$	$term_2$	$MCSA$	ics_{Lin}
choose#1	select#1	%synonyms%	1.0
group#1	category#1	group#1	0.58
product#1	product#1		1.0
$SemSim(l, c)$			0.88

Table 4.6: Semantic similarity measure between an activity label and an ontology concept

Ontology Concept	$SemSim(l, c)$
to_select_product_category	0.88
to_select_product_quantity	0.83
to_select_category	0.74
to_select_quantity	0.7
to_select_method_payment	0.62

Table 4.7: Five ontology concepts most similar to the label "Choose a product group"

Let us consider, for example, the semantic similarity of the label "Choose a product group" and the semantic concept to_select_product_category. In this case, the label and the concept contain a verb, an object and an object specifier, which are easily matched ($CSP = \{$(choose, select), (group, category), (product, product)$\}$). We weight these three different linguistic components according to the proportions: 4:2:1 (in the algorithm of Figure 5 V_WEIGHT, OBJ_WEIGHT and OBJ_SPEC_WEIGHT, respectively). Hence, the formula for the semantic similarity becomes: $SemSim(l, c) = (4 * ics_{Lin}(verb_l, verb_c) + 2 * ics_{Lin}(object_l, object_c) + ics_{Lin}(objSpec_l, objSpec_c))/7$, where l is the label and c is the ontology concept. Table 4.6 shows the result for this pair.

Once the semantic similarity measure is known for all pairs, consisting of a BPMN element label and a BDO concept, we determine the subset of such pairs which maximizes the total semantic similarity, using the maximum cut algorithm [37] applied to the bipartite graph of BPMN element labels and ontology concepts. The result is a suggested semantic annotation for each BPMN element.

Table 4.7 shows the five highest values of semantic similarity between the label "Choose a product group" and each of the concepts in a manually built ontology. The highest score determines the concept automatically suggested to annotate the task labelled "Choose a product group". This algorithm is specialized for the automatic annotation suggestion of process activities but it can be extended to other types of BPMN elements, e.g., data objects (by exploiting their labels) or swimlanes (by exploiting their names), by slightly adapting, in these cases, the clause analysis and the weight assignment.

4.2.4 Domain Ontology Extension

Often available domain ontologies are not directly applicable for the annotation of a specific business process, which may need concepts narrowed to its own, specific domain. We propose a semi-automatic approach to suggest new concepts, missing in the available ontology, so as to support business designers in the BDO extension. At the same time, we want to avoid as much as possible term redundancy. For domain extension, we exploit the linguistic information carried by process labels, by considering only its main clause and, within the main clause, its main components (verbs, objects and object specifiers).

Ontology extension consists of the addition of new concepts, which can be either completely new in the ontology or can be obtained by combining concepts already in the ontology. Let us start with considering the second case. The name of the new concept is a compound name created from other concept names according to the following heuristic rules (summarized in Table 4.8):

($H1$) the new concept is a sub-concept of the concept whose name is the head word of the sentence;

($H2$) if the compound name of the new concept combines a verb and an object concept, it will be a sub-concept of the verb concept;

($H3$) if the compound name of the new concept combines a specified noun and a specifier noun concept, it will be a sub-concept of the specified noun concept;

($H4$) if the verb (the specified noun) in the compound verb-object (specified-specifier) name of the new concept appears in the ontology in the form of the combination of the verb with another object (of the specified noun with another specifier), the verb (the specified noun) is added to the ontology as a single word and a new is_a relationship is added between the verb (the specified noun) and both the old and the new compound concepts. Algorithm 6 shows the pseudo-code of the algorithm implementing these heuristics.

Let us consider, for example, the label "Choose a product" and let us assume that the ontology already contains a to_select concept, whose semantic similarity with the verb "choose" is 1.0; a concept good, whose semantic similarity with the word "product" is 0.45; and no concept exists with an acceptable semantic similarity for the whole label. The two concepts to_select and good can be composed, thus generating a new concept to_select_good (that will be a subconcept of the concept to_select), whose similarity value, with respect to the "Choose a product" label, is 0.92.

There are also cases in which new concepts, made of single words, have to be introduced in the ontology. In order to provide a quite flexible way for managing such a situation we introduce the possibility to specify two thresholds t_w and t_s. The former is referred to

Id	Linguistic Analysis	Ontology
H1	$v \in WS$; $GCat(v) = V$	$v' = SynsetRepr(v)$: v' *is_a Action*
H2	$n \in WS$; $GCat(n) = N$	$n' = SynsetRepr(n)$: n' *is_a Object*
H3	$v \in WS$; $GCat(v) = V$ $o \in WS$; $GCat(o) = N$ v *obj* o	$v' = SynsetRepr(v)$: v' *is_a Action* $o' = SynsetRepr(o)$: o' *is_a Object* $v'o'$ *is_a* v', $v'o'$ *hasTargetObject* o'
H4	$n \in WS$; $GCat(n) = N$ $s \in WS$; $GCat(s) = N$ n *nn* s	$n' = SynsetRepr(n)$: n' *is_a Object* $s' = SynsetRepr(s)$: s' *is_a Object* $s'n'$ *is_a* n', $s'n'$ *hasObjectSpecifier* s'
H5	$v \in WS$; $GCat(v) = V$ $o \in WS$; $GCat(o) = N$ $s \in WS$; $GCat(s) = N$ v *nn* o o *nn* s	$v' = SynsetRepr(v)$: v' *is_a Object* $o' = SynsetRepr(o)$: o' *is_a Object* $s' = SynsetRepr(s)$: s' *is_a Object* $v'o'$ *is_a* v', $v'o'$ *hasTargetObject* o' $s'o'$ *is_a* o', $s'o'$ *hasObjectSpecifier* s' $v's'o'$ *is_a* $v'o'$, $v's'o'$ *hasTargetObject* $s'o'$

Table 4.8: Heuristic rules to create a candidate ontology

Algorithm 6 compose

```
Input $(v, o, os)$: triple of verb, object and object specifier composing a sentence
Input $DO$: domain ontology
Output $DO'$: enriched domain ontology
Output $comp\_sen$: composed sentence
 1: $DO' \leftarrow DO$
 2: if $o \neq null \wedge os \neq null$ then
 3:     $oos \leftarrow$ concatenate $os$ to $o$
 4:     if $oos \notin classes(DO')$ then
 5:         $DO' \leftarrow$ add $oos$ as $o$'s child
 6:     end if
 7:     $comp\_sen = oos$
 8: end if
 9: if $v \neq null \wedge o \neq null$ then
10:     $vo \leftarrow$ concatenate $o$ to $v$
11:     if $vo \notin classes(DO')$ then
12:         $DO' \leftarrow$ add $vo$ as $v$'s child
13:     end if
14:     $comp\_sen = vo$
15:     if $oos \neq null$ then
16:         $voos \leftarrow$ concatenate $os$ to $o$
17:         if $voos \notin classes(DO')$ then
18:             $DO' \leftarrow$ add $voos$ as $vo$'s child
19:         end if
20:         $comp\_sen = voos$
21:     end if
22: end if
23: return$\langle DO', comp\_sen \rangle$
```

single words, i.e., it allows discarding a matching between a single word and an ontology concept whose semantic similarity value is under the threshold. The latter, instead, is a parameter referring to whole sentences, i.e., it allows to determine the set of pairs (label, ontology concept) having an acceptable global similarity, thus considering these concepts as good candidate annotations for the given process activity label.

Whenever no ontology concept (directly contained in the ontology or composed of other ontology concepts) reaches the t_s threshold, a new concept is added to the domain ontology. In order to decide which sentence component (i.e., the verb, the object or the object specifier) to add, we follow the ranking given to the various parts of a sentence (verb: 4; object: 2; specifier: 1). If the label contains a verb that the concept with the highest similarity does not contain, we add the verb; otherwise we repeat the same check for the object and, eventually, in case of failure, for the object specifier. If the sentence threshold t_s cannot be satisfied, we have to add one or more concepts to the ontology so as to be able to match the words in the label. We give precedence to the missing concept with the lowest semantic similarity value among those characterizing each sentence component, so as to increase the similarity of the least similar label component (i.e., the component with the highest margin of similarity improvement). Algorithm 7 shows the pseudo-code for this choice mechanism.

The introduction of a new concept in the ontology raises the problem of its relationships with the concepts already in the ontology. We limit our analysis to the identification of the child-parent relationship. To this purpose, we again exploit the properties of Lin's semantic similarity. For each possible parent concept pc in the hierarchical structure of the ontology (i.e., for each possible direct super-concept), restricted to the concept type category, we compute $ics_{Lin}(\overline{w}_{ca}, RC_{pc}(ca))$, i.e., the average of the maximum semantic similarity values over the possible senses of the head word of the new concept, $\overline{w}_{ca}$, between $\overline{w}_{ca}$ and the head word of each of the relatives in $RC_{pc}(ca)$, that the concept would have if it were a sub-concept of pc. The highest similarity value with respect to all the possible parent concepts, determines the direct super-concept. Algorithm 8 reports the pseudo-code of the described procedure.

Let us consider the ontology in Figure 4.9 and a process activity labelled "Choose a product group". Let us assume word and sentence thresholds equal to 0.6 and 0.8, respectively. Since none of the possible combinations of concepts already in the ontology (those with the highest values are shown in Table 4.10) allows to satisfy the two thresholds ($ics_{Lin}(group\#1, event\#1) < t_w$, $ics_{Lin}(group\#1, family\#2) < t_w$ and all the other concept combinations have lower values), the group concept (in the ranking verb-object-object

Algorithm 7 sentence_suggestion

```
Input $l$: activity label
Input $t_w$: word threshold
Input $t_s$: sentence threshold
Input $DO$: domain ontology
Output $DO'$: enriched domain ontology
Output $sugg$: sentence suggestion for the semantic annotation of the label $l$
 1: $DO' \leftarrow DO$
 2: $SemSim = max_{c_i \in classes(DO') | \langle CSP_i, SemSim_i \rangle = analyze\_clauses(l, c_i)} SemSim_i$
 3: while $SemSim < t_s$ do
 4:     if $V(l) \neq \emptyset$ then
 5:         $v_l \in V(l) | v_l$ is higher in the parsing tree
 6:         $v_c = arg_{max}(max_{c_i \in classes(DO')} ics_{Lin}(v_l, c_i))$
 7:         if $ics_{Lin}(v_l, v_c) < t_w$ then
 8:             $DO' \leftarrow add\_new\_concept(v_l, DO')$
 9:             $v_c = v_l$
10:         end if
11:         $SemSim = VERB\_WEIGHT * ics_{Lin}(v_l, v_c)$
12:         if $SemSim > t_s$ then
13:             return $\langle DO', v_c \rangle$
14:         end if
15:     end if
16:     if $o(v_l)$ exists then
17:         $o_l = o(v_l)$
18:         $o_c = arg_{max}(max_{c_i \in classes(DO')} ics_{Lin}(o_l, c_i))$
19:         if $ics_{Lin}(o_l, o_c) < t_w$ then
20:             $DO' \leftarrow add\_new\_concept(v_l, DO')$
21:             $o_c = o_l$
22:         end if
23:         $SemSim = SemSim + OBJ\_WEIGHT * ics_{Lin}(o_l, o_c)$
24:         if $SemSim > t_s$ then
25:             return $compose(v_c, o_c, null)$
26:         end if
27:     end if
28:     if $s(o_l)$ exists then
29:         $os_l = s(o_l)$
30:         $os_c = arg_{max}(max_{c_i \in classes(DO')} ics_{Lin}(os_l, c_i))$
31:         if $ics_{Lin}(os_l, os_c) < t_w$ then
32:             $DO' \leftarrow add\_new\_concept(os_l, DO')$
33:             $os_c = os_l$
34:         end if
35:         $SemSim = SemSim + OBJ\_SPEC\_WEIGHT * ics_{Lin}(os_l, os_c)$
36:         if $SemSim > t_s$ then
37:             return $compose(v_c, o_c, os_c)$
38:         end if
39:     end if
40: end while
41: $(w_l^*, w_c^*) = arg_{min}(min_{(w_l, w_c) \in \{(v_l, v_c), (o_l, o_c), (os_l, os_c)\}} ics_{Lin}(w_l, w_c))$
42: $DO' \leftarrow add\_new\_concept(w_l^*, DO')$
43: return $(DO', null)$
```

Algorithm 8 add_new_concept

```
Input $DO$: domain ontology
Input $ca$: new concept to add to the ontology $DO$
Output $DO'$: enriched domain ontology
 1: $DO' \leftarrow DO$
 2: $\overline{w}_{ca} = head_s(ca)$
 3: $\overline{w}_{ca}\_type = wnPOS(mGCat(\overline{w}_{ca}))$
 4: for each $pc \in classes(DO')$ do
 5:   $\overline{w}_{pc} = head_s(pc)$
 6:   $\overline{w}_{pc}\_type = wnPOS(mGCat(\overline{w}_{pc}))$
 7:   if $\overline{w}_{pc}\_type == \overline{w}_{ca}\_type$ then
 8:     $RC_{pc}(ca) = CC(pc) \cup \{pc\}$ (i.e., ca's relatives if it were pc's son leaf)
 9:     $rc_n = 0$
10:     for each $rc \in RC_{pc}(ca)$ do
11:       $\overline{w}_{rc} = head_s(rc)$
12:       $\overline{w}_{rc}\_type = wnPOS(mGCat(\overline{w}_{rc}))$
13:       $\overline{s}_{rc} = get\_sense(rc)$
14:       if $\overline{w}_{rc}\_type == \overline{w}_{ca}\_type$ then
15:         $maxics(\overline{w}_{ca}, \overline{w}_{rc}) = max_{s_i \in senses(\overline{w}_{ca}, \overline{w}_{ca}\_type)} ics_{Lin}((\overline{w}_{ca}, s_i), (\overline{w}_{rc}, \overline{s}_{rc}))$
16:         $rc_n = rc_n + 1$
17:       end if
18:     end for
19:     $avgics(\overline{w}_{ca}, RC_{pc}ca) = \frac{\sum_{rc \in RC_{pc}(ca)} maxics(\overline{w}_{ca}, \overline{w}_{rc})}{rc_n}$
20:   end if
21: end for
22: $p_c^* = arg_{max}(max_{pc_i \in classes(DO')} avg(\overline{w}_{ca}, RC_{pc}(ca)))$
23: $DO' \leftarrow$ add $ca$ as $pc$'s child
24: $returnDO'$
```

Concept position p (direct sub-concept of)	$ics(\overline{w}_{ca}, RC(p))$
family	0.76
social-object	0.61
computer-data	0.45
user-name	0.41
abstract-object	0.38

Table 4.9: Ranking of BDO concepts similar to the new concept group (to be added to the ontology)

specifier, the first part of the sentence with $ics_{Lin} < t_w$) needs to be added.

Table 4.9 reports the semantic similarity values $ics(\overline{w}_{ca}, RC_{pc}(ca))$ for the head word $\overline{w}_{ca}$ = "group". A direct is_a relationship with the family concept, corresponding to the best value, is suggested to the designer.

Let us consider again the label "Choose a product group" and let us now suppose that this is the first label we are going to analyse (i.e., the skeleton ontology is still empty). By MINIPAR, we obtain the information in Figure 4.7 (top left). According to the heuristics $H1$, $H2$, the concept to_choose is added to the ontology skeleton as an Action sub-concept, and product and group concepts as Object sub-concepts. Since the word "group" is the object of the verb "choose", we can apply $H3$ and build the sub-concept to_choose_product, whose has_target_object relationship is restricted to the concept group. Since the word "product" specifies the word "group", product_group is added as group

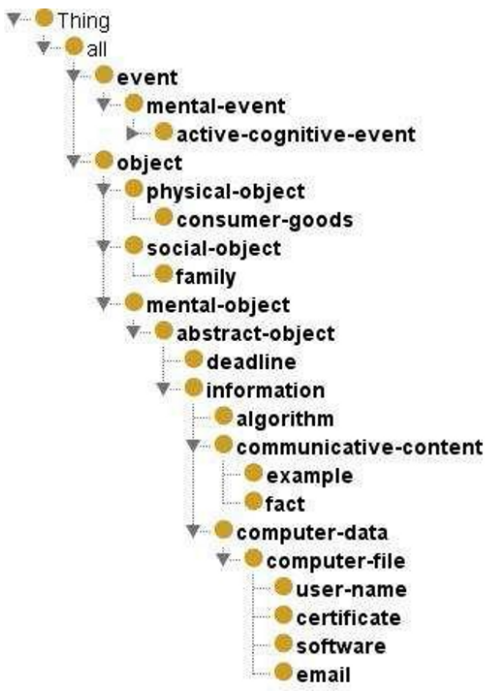


Figure 4.9: Automatically suggested position for concept group

Label word w_1	**Concept word** w_2	ICS_{Lin} (w_1, w_2)	**Activity label**	**Concept name**	$SemSim(l, c)$
Choose#1 group#1 product#1	to_select#1 event#1 good#4	1.0 0.5 0.76	Choose a product group	to_select_good_event	0.823
Choose#1 group#1 product#1	to_select#1 family#2 good#4	1.0 0.47 0.76	Choose a product group	to_select_good_family	0.815
Choose#1 group#1 product#1	to_determine#1 event#1 good#4	0.97 0.5 0.76	Choose a product group	to_determine_good_event	0.82

Table 4.10: Three composed concepts most similar to the label “Choose a product group”

sub-concept, with has_object_specifier object property restricted to the concept product (heuristics $H4$). The concept to_choose_product_group, whose has_target_object property is restricted to product_group, is added as a sub-concept of to_choose_group. The resulting ontology fragment is shown in Figure 4.10 (left).

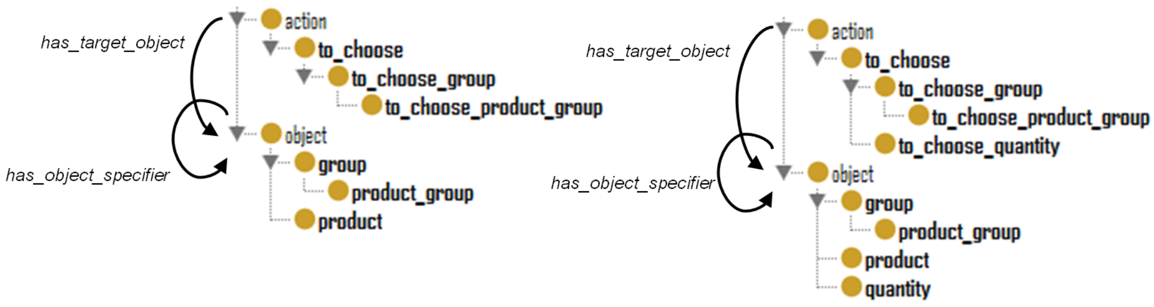


Figure 4.10: Candidate ontology skeleton construction

For the label “Select quantity” MINIPAR suggests (top right in Figure 4.7) that “select” is a verb and “quantity” a noun. Since the ontology already contains the concept to_choose, returned by *SynsetRepr* as the canonical representative of the “select” synset, to_select is not added to the ontology. The concept quantity is added as an Object sub-concept and the concept to_choose_quantity as a sub-concept of the concept to_choose, with the has_target_object property restricted to quantity. The resulting updated ontology structure is shown in Figure 4.10 (right).

4.2.5 Automatic Suggestion Evaluation

In this subsection we provide a first evaluation of the proposed techniques, both those analysing the domain ontology for term disambiguation and those supporting designers with the automated suggestion of semantic annotations and ontology extension, when necessary.

Domain Ontology Analysis

The three algorithms for sense disambiguation of head words of domain ontology concepts have been applied to three small ontologies (selected portions of OntoSem).

The first ontology, shown in Figure 4.11, is a generic ontology, classifying terms into very high-level categories. These categories have a structural organization quite different from the one adopted in WordNet, in which the hierarchies of different grammatical

$head_s(w)$	RDA sense	RDA $semSim$	CDA sense	CDA $semSim$	DBDA sense	DBDA $semSim$	Correct sense
event	-	0.0	-	0.0	-	0.0	-
analyze	#1	0.5	-	0.0	#1	0.11	#1
determine	#6	0.24	#5	0.22	#3	0.5	#3
compare	#3	0.33	#3	0.5	#1	0.75	#1
differentiate	#3	0.5	#3	0.5	-	0.0	#1
decide	#1	0.23	-	0.0	-	0.0	#1
err	-	0.0	-	0.0	-	0.0	#1
define	#1	0.31	#5	0.9	#3	0.4	#3
evaluate	#2	0.4	-	0.0	-	0.0	#1
identify	#3	0.33	-	0.0	#1	0.5	#1
solve	#2	0.32	-	0.0	#1	0.29	#1
test	#1	0.34	#6	0.57	#1	0.11	#5
study	#1	0.5	-	0.0	-	0.0	#1

Table 4.11: Results obtained by applying the disambiguation algorithms to the ontology in Figure 4.11

categories are strictly separated. By applying the three disambiguation algorithms to this ontology we get the results presented in Table 4.11.

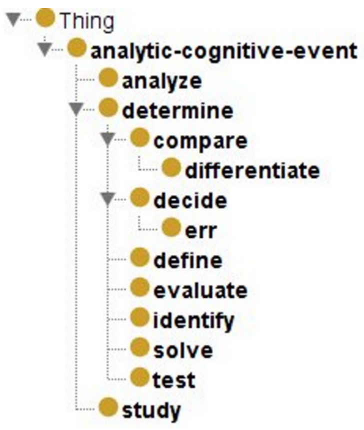


Figure 4.11: An example of an ontology with a structure different from the one of WordNet

The RDA and CDA algorithms seem not to work very well: only 3 senses out of 12 are correct for the first algorithm and none for the second. For both algorithms, the similarity values guiding the choice of the sense are lower than 0.5 (except for the concept define in the CDA), indicating that the match found by the algorithm has low confidence and that it would be highly recommended to complement both algorithms with the outcome of the DBDA algorithm. Indeed, the DBDA algorithm has better performance: 7 out of 12 correct senses; moreover, in case of incorrect or absent answers it also reports a very low similarity value, indicating low confidence in the match that was found. Hence, in this case, the similarity level provides an important clue regarding the reliability of the results.

$head_s(w)$	RDA sense	RDA $semSim$	CDA sense	CDA $semSim$	DBDA sense	DBDA $semSim$	Correct sense
agree	#2	0.4	#2	0.4	#2	0.22	#2
approve	#1	0.4	#1	0.8	#1	0.2	#1
promise	#1	0.43	#1	0.86	#1	0.14	#1
guarantee	#1	0.86	#1	0.9	#1	0.08	#1
swear	#3	0.82	#3	0.82	#4	0.11	#3

Table 4.12: Results obtained by applying the disambiguation algorithms to the ontology in Figure 4.12 (left)

$head_s(w)$	RDA sense	RDA $semSim$	CDA sense	CDA $semSim$	DBDA sense	DBDA $semSim$	Correct sense
push	#1	0.65	#1	0.37	#1	0.5	#1
expel	#1	0.53	#1	0.6	#1	0.2	#1
launch	#2	0.5	#2	0.53	#2	0.2	#2
press	#5	0.24	#5	0.5	#5	0.1	#5
imprint	#2	0.5	#2	0.5	#2	0.2	#2

Table 4.13: Results obtained by applying the disambiguation algorithms to the ontology in Figure 4.12 (right)

On the contrary, when considering ontologies with a structure more similar to the WordNet one and characterized by a sufficient level of detail, as in the case of the ontologies in Figure 4.12, the performance of the disambiguation algorithms improves substantially. As shown in Table 4.12 and Table 4.13, both the RDA and the CDA algorithms provide very good results corresponding to high similarity values (almost all are greater than 0.4), while DBDA performs just a bit worse than the other algorithms (giving very low similarity values).

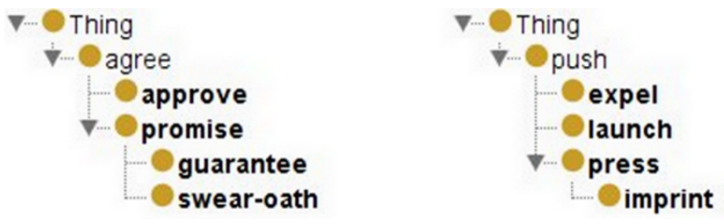


Figure 4.12: Two examples of ontologies with structure and granularity similar to the WordNet one

These results suggest therefore the possibility of integrating the RDA and the CDA algorithms with the DBDA one, when the domain ontology comes with descriptions that can be compared to those that can be found in WordNet. The choice of the algorithm can be done on the basis of the ontology type (e.g., considering its similarity with the WordNet structure and its granularity level), whenever the problem due to different grammatical categories belonging to the same hierarchy occurs, but also by evaluating the similarity values independently for each word.

Semantic Annotation Suggestion and Domain Ontology Extension

The approaches proposed for the business process semantic annotation and ontology extension have been applied to an on-Line Shop process and to an extract of a generic ontology (both available at `http://selab.fbk.eu/OnLineShop`). The process contains 36 activities to be annotated. The ontology, instead, is an extract of 231 classes out of the 7956 of the OntoSem ontology, whose concepts have been previously mapped to WordNet synsets using our approach.

For this case study, we chose as thresholds for word and sentence acceptances $t_w = 0.6$ and $t_s = 0.85$, respectively, based on our previous experience in similar annotation exercises. When considering the first label, "Choose a product group", the business designer is suggested to extend the ontology with the concept group. In fact, the maximum similarity value of the label with a concept in the ontology, is obtained with the to_select concept and is equal to 0.57, i.e., the weighted average over the semantic similarity values of the sentence components (1.0/(4+2+1) = 0.57), which is below t_s. Moreover, introducing new concepts by composing concept names already in the ontology is also not enough for satisfying the threshold: the best information content similarity value for the object "group" is $0.5 < t_w$; for the sentence it is $0.68 < t_s$. A group concept has therefore to be added to the ontology, as well as the composed concepts to_select_group, good_group and to_select_good_group, since the word "product" is matched with the concept good in the maximized information content similarity.

When analysing the label "Update product quantity" the concept product is proposed as a new concept for the ontology, since its best information content similarity value with respect to the ontology concepts is lower than those characterizing the other parts of the sentence. After the business designer accepts this suggestion, it is possible to annotate the current activity with a new to_modify_product concept. It is also possible to improve the annotation of the "Choose a product group" activity: the annotation becomes to_select_product_group and the new similarity value 1.0. Whenever the ontology is extended, the previous annotation suggestions are automatically revised to identify cases where a better match has become possible.

Going ahead with the annotation of the other activities, the process will finally be annotated and the ontology extended with new concepts. The automatically suggested annotations are shown in the second column in Table 4.14. The single words added as new concepts to the ontology are marked by an asterisk. On the contrary, since the starting

Activity label	**Automated suggestion**	**S.sim**	**Manual annotation**	*R&C*	*R*	*TBR*
Choose a product group	to_select_product*_group*	1.0	to_select_product_group	3	3	3
Search for a product		0.0	to_search_for_product	0	0	2
Read policies	to_read_document	0.9	to_read_policy	1	2	2
Choose a product	to_select_product*	1.0	to_select_product	1	2	2
Select quantity	to_select_number	0.93	to_select_number	2	2	2
Add the product to the cart	to_add_product*	1.0	to_add_product	2	2	2
Update product quantity	to_modify_product*_number	0.86	to_modify_product_number	3	3	3
Remove product from cart	to_remove_product*	1.0	to_remove_product	2	2	2
Ask for checkout	to_request_checkout*	0.97	to_request_checkout	2	2	2
Provide personal data	to_supply_data	1.0	to_supply_data	2	2	2
Log-in	to_logIn*	1.0	to_login	1	1	1
Choose shipment method	to_select_product*_method	0.98	to_select_shipment_method	2	3	3
Choose a payment method	to_select_marketing_method	0.96	to_select_payment_method	2	3	3
Provide payment information	to_supply_marketing_data	0.96	to_supply_payment_data	2	3	3
Confirm order	to_confirm_order	1.0	to_confirm_order	2	2	2
Show the home page	to_show_page*	0.86	to_show_home_page	2	2	3
Provide summarized product info	to_supply	1.0	to_supply_product_data	1	1	3
Search for a product		0.0	to_search_for_product	0	0	2
Provide policy information	to_supply_document_data	0.95	to_supply_policy_data	2	3	3
Show product data	to_show_product*_data	1.0	to_show_product_data	3	3	3
Provide detailed product information	to_supply_product*_data	1.0	to_supply_product_data	3	3	3
Check product quantity availability	to_confirm_product*_availability*	0.96	to_check_product_availability	2	3	3
Create cart	to_create_cart*	1.0	to_create_cart	2	2	2
Warn buyer	to_warn*_buyer*	1.0	to_warn_buyer	2	2	2
Compute total	to_calculate_model	0.86	to_calculate_total	1	2	2
Visualize cart	to_visualize*_cart*	1.0	showCart	1	2	2
Check out	to_confirm	0.93	to_check_out	0	1	1
Collect personal data	to_accumulate_data	1.0	to_accumulate_data	2	2	2
Check login data		0.0	to_check_login_data	0	0	3
Store shipment method		0.0	to_store_shipment_method	0	0	3
Store payment method		0.0	to_store_payment_method	0	0	3
Store payment information		0.0	to_store_payment_data	0	0	3
Update stocked product data	to_modify_product*	0.92	to_modify_product_data	3	3	3
				52	61	80

Table 4.14: Case study data for the semi-automatic annotation suggestion and domain ontology extension

ontology does not contain composite concepts of the form verb-object, specified-specifier, verb-specifier-object, almost all the composite concepts have been automatically added to the ontology during the preliminary analysis of the process labels.

Without suggestions, the semantic annotation activity is hard, mainly in the case of huge domain ontologies. As an example, let us consider the activity labelled with the short sentence "Provide product information". In order to find a good annotation for the verb "provide" (i.e., the concept supply), the business designer has to go down, by manually browsing the extract of the OntoSem ontology, through four nested levels from the event category (Figure 4.13, left). Moreover, while browsing the ontology, she could also be wrong by choosing as semantic annotation the verb give instead of supply. This verb, in fact, may seem to be a synonym of the verb "to_provide", while in OntoSem it represents the transfer of possession. Similarly, before retrieving the noun data for the semantic annotation of the "information" part of the short sentence, the business designer has to go down through 3 levels of nesting (Figure 4.13, right). Also in this case, she may easily be wrong, maybe stopping at a higher nesting level and therefore choosing a too general concept (for example knowledge instead of data). Finally in the case of the term "product" a new term has to be added to the ontology and, before coming up with this resolution, the business designer needs to browse a relevant part of the ontology to check whether the needed concept exists or not.

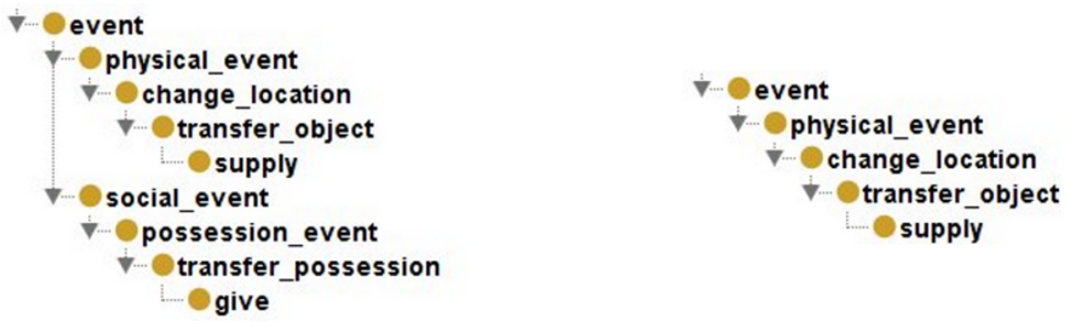


Figure 4.13: Extracts of the OntoSem ontology

In order to evaluate the approach, we asked a human (a BPMN expert with a good domain knowledge) to perform the same task, starting from the same domain ontology and giving her the possibility to add new concepts, when necessary, but working without any automated suggestion. The guidelines followed in the execution of this exercise were similar to those implemented in our approach. We compared (Table 4.14) the manual semantic annotations, i.e., our gold standard, with those obtained with the automated approach. The comparison has been performed separately on the three main label components (i.e., verb, object, and object specifier). We base the evaluation on two assumptions:

1. if a concept for the annotation of a part of a sentence is in the ontology, in order to be correct, it has to be exactly the same in both manual and automated result;
2. if a new concept has to be added to the ontology it will likely have the same name of the sentence part it is required to match.

We define: (1) reported and correct ($R\&C$), the number of label parts semantically annotated by our technique with exactly the same concepts used in the gold standard; (2) reported (R), the number of the label parts for which our technique has provided a suggestion; (3) to be reported (TBR), the number of annotation concepts (already in the ontology or added later) in the gold standard. We computed precision and recall for our case study: $precision = R\&C/R = 0.85$ and $recall = R\&C/TBR = 0.65$.

Both these quantitative results and the qualitative assessment we made of the results shown in Table 4.14 indicate that the proposed approach is viable and effective in supporting the difficult and expensive task of adding semantic annotations to business processes.

Chapter 5

Constraint Verification

“Contrariwise, if it was so, it might be;
and if it were so, it would be; but as it isn’t, it ain’t.”
Lewis Carrol

A crucial step in process modelling is the creation of valid diagrams, which not only comply with the basic requirements of the process semantics, but also satisfy properties associated with the specific process domain. For instance, an important requirement for a valid on-line shopping process should be the fact that *the activity of providing personal data is always preceded by an activity of reading the policy of the organization.*

As the notion of semantically annotated processes becomes more and more popular (e.g., [167], [54]) and business experts start annotating elements of their processes with semantic objects taken from a domain ontology, there is an increasing potential to use the Semantic Web technology to support business experts in their modelling activities, including the modelling of valid diagrams which satisfy semantically enriched and domain specific constraints. In turn, the same process annotation can be constrained by special requirements. An example of the latter constraints in the on-line shopping process is the fact that *a complex action like managing the cart cannot be used for annotating an atomic BPMN task.*

Our proposal of enriching process models with concepts belonging to (a set of) domain ontology(es) (described in Chapter 4) aims at supporting business designers also in constraint verification. The domain semantic information added to the process elements as well as the formalization of semantically annotated processes into the BPKB, in fact, enable automated reasoning on the process and on its constraints, thus supporting business experts in the realization of valid process models.

In this chapter, we propose a concrete formalization of typical classes of structural requirements over annotated BPMN processes in Description Logics [12] (Section 5.1) and we show how Description Logic (DL) reasoners can be used to provide the automated verification of process requirements and models (Section 5.2). Finally, we provide an evaluation of the proposed approach in terms of DL expressivity and performance (Section 5.3).

The material presented in this chapter has been published in [44, 45, 46, 153].

5.1 Process Requirement Specification

Many recent works in the literature use semantic annotations for supporting process modelling activities. These different approaches can be classified in two main groups: (i) those adding semantics to specify the dynamic behaviour exhibited by a business process, and (ii) those adding semantics to specify the meaning of the entities of a business process in order to improve the automation of business process management.

Our work falls in the second group and, in detail, it focuses on the usage of Semantic Web technology to specify and verify structural constraints, that is, constraints that descend from structural requirements which refer to descriptive properties of the annotated process diagram and not to its execution. The reason for this choice is twofold: (i) structural requirements complement behavioural properties, as they can be used to express properties of the process which cannot be detected by observing the execution of a process; (ii) structural requirements provide an important class of expressions whose satisfiability can be directly verified with existing DL reasoners.

In order to provide examples of structural requirements and clarify how they are enforced in the process, we describe an on-line shopping process and a set of requirements that could be specified by business experts on the process itself in the next subsection (Subsection 5.1.1). In detail, to verify the process adherence to the specified requirements, we transform them into constraints. As already described in Subsection 4.2.5 of Chapter 4, we distinguish between two different kinds of constraints: **merging axioms** and **process specific constraints**. In Subsection 5.1.2 and 5.1.3 we provide details about merging axioms and process specific constraints, respectively.

5.1.1 An Explanatory Example

We use the portion of a semantically annotated on-line shopping process reported in Figure 5.1 as an explanatory example in the whole section. The process represents the initial steps of an on-line shopping process (e.g., the product presentation and selection

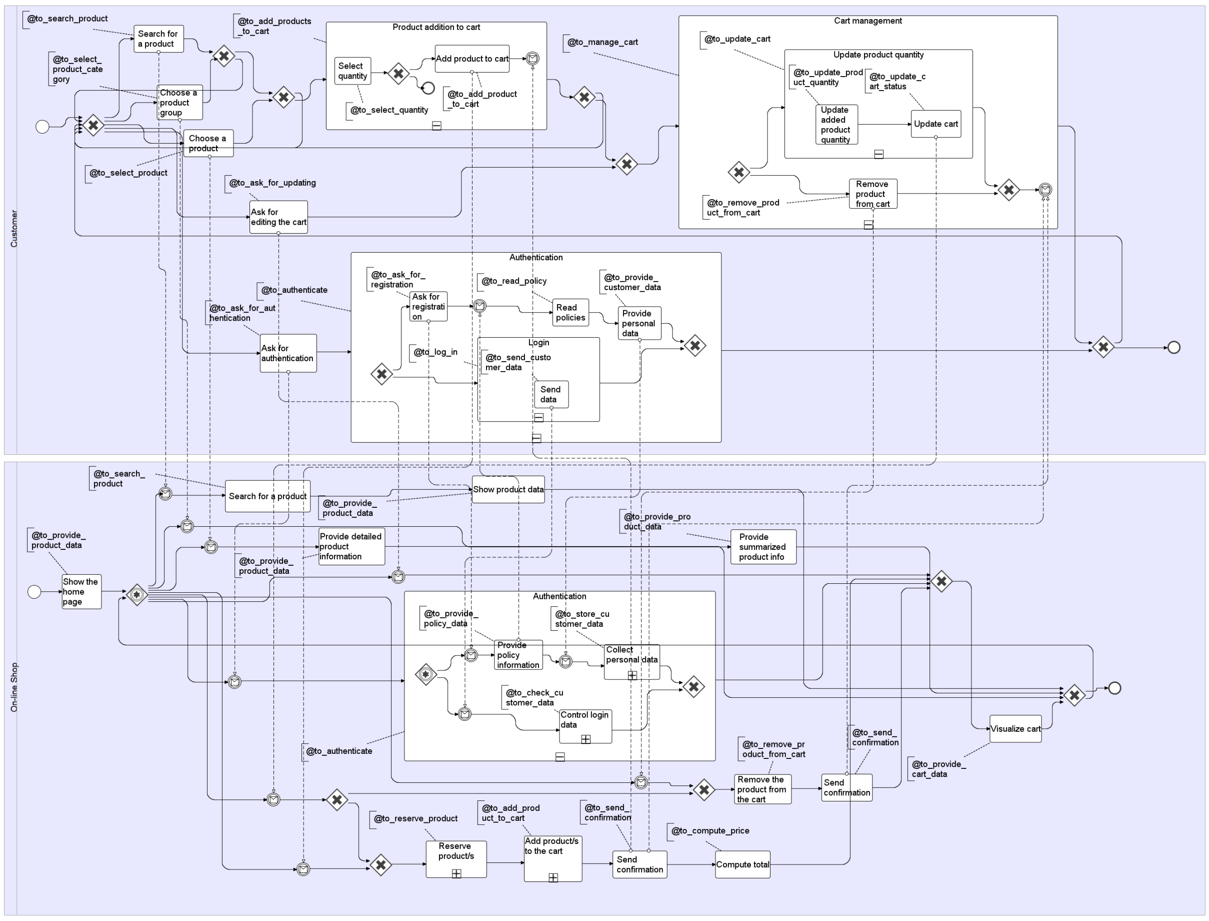


Figure 5.1: A portion of the on-line shopping business process diagram.

and the customer authentication), leaving out the last phases, e.g., the checkout. It is structured in two sides: the server side, represented by the On-line Shop pool, which describes the process from the point of view of the shop, and the client side, represented by the Customer pool, describing the process from the point of view of the buyers.

The realization of the on-line shopping process depicted in Figure 5.1 can involve a team of business experts, who may wish to impose requirements (constraints) on the process itself. These requirements could cover different aspects of the process, ranging from the correct annotation of business process elements to security issues, from privacy issues to issues related to management of exceptions and exception handling mechanisms, as in the following examples:

- issues related to the semantic annotation:

  (a) *"to_manage" is a complex action and can be used only to annotate BPMN sub-*

processes (and not atomic activities).

- privacy issues:

  (b) *the activity of providing personal data is always preceded by the activity of reading the policy of the organization*;

  (c) *the activity of reading the policy of the organization is activated by an event generated from the activity of providing these policies to the customer itself*;

- security issues:

  (d) *the Customer pool must contain an authentication sub-process which, in turn, contains a log-in activity and an insertion of personal data activity*;

- issues related to the exception handling:

  (e) *the activity of reserving products in the On-line Shop pool has always to catch a "product unavailability" error event*;

  (f) *the "product unavailability" error event caught by the activity of reserving products in the On-line Shop pool has to be handled by executing in parallel two activities. The first one is an activity for warning the buyer; the second one is a sub-process for ordering the unavailable products*;

  (g) *the activity of "sending customer data" in the "log-in" sub-process has always to allow, after its execution, receiving a "compulsory log-in failure" error event from the On-line Shop pool. The "log-in" sub-process has, in turn, always to catch this error and the error event has to be handled by immediately stopping the process*;

- general issues :

  (h) *in the on-line shopping process there must be a Customer pool and an On-line shop pool*;

  (i) *inclusive gateways cannot be used in the on-line shopping process (to force all the alternative actions to be mutually exclusive)*;

  (j) *each gateway must have at most 2 outgoing gates (to keep the process simple)*;

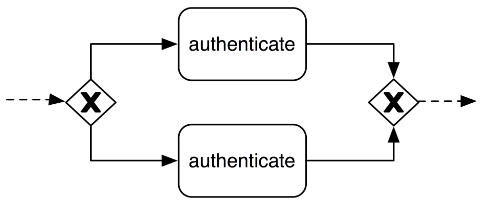
  (k) *each pool must contain a single authentication activity / sub-process (to ease maintenance)*;

(l) *the activity of managing a shopping cart is a sub-process which contains an activity removing products from the cart.*

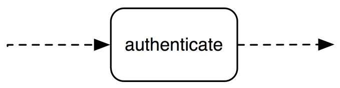
All these constraints are examples of structural requirements as they are related to the descriptive properties of the annotated process diagram and complement properties which may refer to the process execution. While some of the requirements listed above can bear some similarity with behavioural properties, it is important to note here that expressing them as structural requirements constraints the structure (in addition to the behaviour), as it is possible to obtain the same process behaviour by completely different process diagrams. To make a simple example we could "rephrase" constraint (k) in the apparently equivalent *the execution paths of all pools must contain a single authentication activity.* Nevertheless, while this requirement is satisfied by both diagrams in Figure 5.2, requirement (k) is only satisfied by diagram 5.2b, which is definitely easier to maintain if changes to the authentication sub-process are foreseen. Thus, structural requirements are the appropriate way to express static properties of the diagram, which may even not be discovered by analysing the behaviour of the process.

5.1.2 Merging Axioms

Process model semantic annotation aims at supporting business experts in design and analysis activities, thus affecting the creation of high quality process models. However an unavoidable precondition to their effectiveness is that they are correct. Though the notion of correct semantic annotation deserves a precise definition, we can intuitively say that a necessary condition for a correct annotation is that it respects types. For example, activities in a business process should be annotated with concepts actually denoting actions; similarly, BPMN data objects should be annotated with concepts representing objects, and so on. Thus, for instance, an activity annotated with a cart concept or a BPMN data-object annotated with a to_manage_cart concept are intuitively incorrect annotations. Additional requirements for a correct annotation could be imposed because of



(a)



(b)

Figure 5.2: Diagrams with equivalent behaviour

the specific application domain. For instance, in a domain in which simple actions only come from a fixed set, BPMN tasks should be only annotated with actions taken from this set. Hence, though belonging to different ontologies, concepts from the BPMN ontology and the domain ontology are not totally unrelated. Precise correspondences which define criteria for correct / incorrect semantic annotations often exist and it is important to make them explicit. Examples of these criteria, which may hold in many application domains, are:

A BPMN activity ***can be annotated only*** *with actions of the domain ontology (and not for example, with objects).* (5.1)

A BPMN data-object ***cannot be annotated*** *with actions or events of the domain ontology (but for example, with objects).* (5.2)

A BPMN Event ***can be annotated only*** *with events of the domain ontology (and not for example, with objects).* (5.3)

A domain specific criterion, which refers to the particular business process or domain ontology at hand, is requirement (a) in Subsection 5.1.1.

To allow the business designer to specify the kind of positive and negative constraints described above, we propose the usage of four constructs: "*annotatable only by*" ($\xrightarrow{\mathsf{AB}}$) and "*not annotatable by*" ($\xrightarrow{\mathsf{nAB}}$) from BPMNO concepts to BDO concepts (also defined as "*can represent only*" and "*cannot represent*", respectively), and the symmetrical "*annotates only*" ($\xrightarrow{\mathsf{A}}$) and "*cannot annotate*" ($\xrightarrow{\mathsf{nA}}$) from BDO concepts to BPMNO concepts (also defined as "*can be represented only as*" and "*cannot be represented as*", respectively). Their intuitive meaning and their formalization as DL axioms is reported in Table 5.1[1]. We use W to denote a concept of BPMNO and Y to denote a concept of BDO.

The formalization of the four constructs as DL axioms is the basis for the translation of informal expressions such as (5.1)–(5.3) and (a) into a formal set of expressions, denoted with $\mathsf{MA}(\mathsf{BPMNO}, \mathsf{BDO})$. Note that though the meaning of $X \xrightarrow{\mathsf{nAB}} Y$ and $Y \xrightarrow{\mathsf{nA}} X$ coincide, we provide both primitives as, depending on the case to be modelled, one may result more intuitive than the other. For example the requirement (a) can be represented in the form $\mathsf{BDO{:}to\_manage} \xrightarrow{\mathsf{A}} \mathsf{BPMNO{:}sub\_process}$, which is formalized as $\mathsf{BDO{:}to\_manage} \sqsubseteq \mathsf{BPMNO{:}sub\_process}$.

Merging axioms can describe "domain independent" criteria, such as (5.1)–(5.3), and "domain specific" criteria, such as requirement (a). Domain independent criteria, may

[1]We recall that we use the term BPMNO-type for specifying the BPMN type of a BPD object, i.e., the BPMNO class/superclass of the corresponding instance in the BPKB, as introduced in Chapter 4.

Merging Axiom	Intuitive meaning	DL Axiom
BPMNO:W $\xrightarrow{\mathsf{AB}}$ BDO:Y	A BPMN element of BPMNO-type W can be annotated only with (can represent only) a domain specific concept equivalent or more specific than Y	BPMNO:W $\sqsubseteq$ BDO:Y
BPMNO:W $\xrightarrow{\mathsf{nAB}}$ BDO:Y	A BPMN element of BPMNO-type W cannot be annotated with (cannot represent) a domain specific concept equivalent or more specific than Y	BPMNO:W $\sqsubseteq \neg$ BDO:Y
BDO:Y $\xrightarrow{\mathsf{A}}$ BPMNO:W	Any domain specific concept equivalent or more specific than Y can be used to annotate only (can be represented only as) (can be represented only as) BPMN elements of BPMNO-type W	BDO:Y $\sqsubseteq$ BPMNO:W
BDO:Y $\xrightarrow{\mathsf{nA}}$ BPMNO:W	Any domain specific concept equivalent or more specific than Y can not be used to annotate (can not be represented as) BPMN elements of BPMNO-type W	BDO:Y $\sqsubseteq \neg$ BPMNO:W

Table 5.1: Merging axiom patterns

hold in many application domains, as they relate elements of BPMN, such as data-objects, activities or events to very general concepts, like the elements of a top-level ontology, e.g., DOLCE [63]. These kinds of constraints can be thought of as "default" criteria for correct / incorrect semantic annotations, and in this case DOLCE can be provided as a "default" component of the domain ontology in the workspace. The advantage of having these criteria already included in the BPKB is that in many situations it might be the case that the analysts, which are usually very focused on their application domain, forget to add them explicitly while they may tend to add more domain-specific constraints; these "default" criteria, however, could still be modified by the analysts to reflect the actual annotation criteria for the specific domain at hand.

To support the creation of merging axioms, a first library of domain independent merging axioms between BPMN and DOLCE has been implemented (see [65] for a detailed description). Based on this work, expression (5.1) can be represented with the merging axiom BPMNO:activity $\xrightarrow{\mathsf{AB}}$ BDO:process (identifying action with class process in DOLCE), which in turn is formally represented with the DL statement BPMNO:activity $\sqsubseteq$ BDO:process, expression (5.2) can be represented with the merging axiom BPMNO:data_object $\xrightarrow{\mathsf{nAB}}$ BDO:perdurant (where DOLCE class perdurant is a general class covering both processes and events) which in turn is represented with BPMNO:data_object $\sqsubseteq \neg$BDO:perdurant, and similarly with the other expressions.

Finally, the specification of these constraints on correct / incorrect annotations can be exploited to refine and strengthen the technique, described in Subsection 4.2.3 of Chapter 4, for the suggestion of candidate semantic annotations. The definition of merging axioms, in fact, allows the suggestion algorithm to both reduce the annotation search space and come up with annotation suggestions compliant with the merging axiom constraints.

5.1.3 Process Specific Constraints

Process specific constraints are expressions used to state specific properties that apply to the process under construction. Differently from merging axioms, these expressions can have many different forms to match different properties of the process. We identified five types of process specific constraints that can be expressed over the Business Process Diagrams: (i) containment constraints (including existence constraints), (ii) enumeration constraints, (iii) precedence constraints, (iv) exception handling constraints and (v) composed constraints.

Containment Constraints

Containment constraints are of the form X *contains* Y or X *does not contain* Y and are used to represent the fact that the BPD or certain graphical elements contain/do not contain other graphical elements. A simple containment constraint of the form X *contains* Y which can be expressed over the on-line shopping process is provided by requirement (1).

Containment constraints can be encoded in Description Logics using specific BPMNO roles which formalise the containment relations existing between different BPD objects as described by specific attributes in [131]. Examples of these roles, used in DL to represent object properties and data properties, are:

- has_embedded_sub_process_sub_graphical_elements, which corresponds to the GraphicalElement attribute of an Embedded Sub-Process, as described in [131], and represents all the objects (e.g., Events, Activities, Gateways, and Artifacts) that are contained within the Embedded Sub-Process;
- has_pool_process_ref, which corresponds to the ProcessRef attribute and is used to represent the process that is contained within a pool;
- has_process_graphical_element, which corresponds to the GraphicalElements attribute of BPMN and identifies all the objects that are contained within a process[2];
- has_business_process_diagram_pools, which allows to relate a BPD with the pools it contains.

For the sake of readability hereafter we abbreviate the four roles described above with has_embedded, has_process_ref, has_graphicals and has_diagram_pools, respectively. Using

[2]We assume that the GraphicalElement attribute of a process identifies all the objects in the process including those nested in embedded sub-processes

r to indicate any of the roles above, containment constraints are typically expressed as statements of the form $X \sqsubseteq \exists r.Y$ or $X \sqsubseteq \forall r.Y$ which use the basic existential and universal quantification constructs $\exists r.Y$ and $\forall r.Y$.

Requirement (l) can therefore be formalized as follows:

$$\mathsf{BDO{:}to\_manage\_cart} \sqsubseteq \mathsf{BPMNO{:}embedded\_sub\_process} \tag{5.4}$$

$$\begin{aligned}\mathsf{BDO{:}to\_manage\_cart} \sqsubseteq \exists \mathsf{BPMNO{:}has\_embedded}.(\mathsf{BPMNO{:}activity} \sqcap \\ \mathsf{BDO{:}to\_remove\_product})\end{aligned} \tag{5.5}$$

Replacing the specific domain activities in requirement (l) with X and Y, we can obtain a pattern of the form *X is a sub-process that contains the activity Y*, which can be formalized as:

$$\mathsf{BDO{:}X} \sqsubseteq \mathsf{BPMNO{:}embedded\_sub\_process} \tag{5.6}$$

$$\mathsf{BDO{:}X} \sqsubseteq \exists \mathsf{BPMNO{:}has\_embedded}.(\mathsf{BPMNO{:}activity} \sqcap \mathsf{BDO{:}Y}) \tag{5.7}$$

Relaxing the condition that X is a sub-process, instead, we obtain the (more general) pattern *The (embedded) sub-process X contains the activity Y*, which is encoded in DL as[3]:

$$\begin{aligned}\mathsf{BDO{:}X} \sqcap \mathsf{BPMNO{:}embedded\_sub\_process} \sqsubseteq \\ \exists \mathsf{BPMNO{:}has\_embedded}.(\mathsf{BPMNO{:}activity} \sqcap \mathsf{BDO{:}Y})\end{aligned}$$

Finally, abstracting also with respect to the BPMNO-type, we obtain a general requirement pattern:

W of BDO-type[4] *X contains Z of BDO-type Y*

The DL formalization of this pattern (and the roles involved in the formalization) changes according to the BPMNO-type W of X. We provide hereafter some of the most common examples:

- In case W is an embedded sub-process or a process, the has_graphicals and has_embedded roles are used in the DL axiom, respectively:

$$\begin{aligned}\mathsf{BDO{:}X} \sqcap \mathsf{BPMNO{:}embedded\_sub\_process/BPMNO{:}process} \sqsubseteq \\ \exists \mathsf{BPMNO{:}has\_embedded/BPMNO{:}has\_graphicals}.(\mathsf{BPMNO{:}Z} \sqcap \mathsf{BDO{:}Y})\end{aligned} \tag{5.8}$$

[3]In order to make the axiom compatible with some of the available tools for encoding axioms, the (non-atomic) expression on the left part of the $\sqsubseteq$ relationship can be reconducted to a single term by defining an auxiliary class. For example, the axiom $expr_1 \sqsubseteq expr_2$ can be reformulated by defining the auxiliary class $aux \equiv expr_1$ and using it in the axiom: $aux \sqsubseteq expr_2$.

[4]We recall that we use the term BDO-type for specifying the domain type of a BPD object, i.e., the BDO class/superclass of the corresponding instance in the BPKB, as introduced in Chapter 4.

- In case W is a pool, the formalization also uses the $\mathsf{has\_process\_ref}$ role:

$$\begin{aligned}&\mathsf{BDO{:}X} \sqcap \mathsf{BPMNO{:}pool} \sqsubseteq \\ &\quad \exists \mathsf{BPMNO{:}has\_process\_ref.}(\mathsf{BPMNO{:}process} \sqcap \\ &\qquad \exists \mathsf{BPMNO{:}has\_graphicals.}(\mathsf{BPMNO{:}Z} \sqcap \mathsf{BDO{:}Y}))\end{aligned} \tag{5.9}$$

- Finally, in case W is a business process diagram (the main diagram or a diagram referenced by a reusable sub-process and encoded in a different BPKB), the formalization also contains the $\mathsf{has\_diagram\_pools}$ role:

$$\begin{aligned}&\mathsf{BDO{:}X} \sqcap \mathsf{BPMNO{:}business\_process\_diagram} \sqsubseteq \\ &\quad \exists \mathsf{BPMNO{:}has\_diagram\_pools.}(\mathsf{BPMNO{:}pool} \sqcap \\ &\qquad \exists \mathsf{BPMNO{:}has\_process\_ref.}(\mathsf{BPMNO{:}process} \sqcap \\ &\qquad\quad \exists \mathsf{BPMNO{:}has\_graphicals.}(\mathsf{BPMNO{:}Z} \sqcap \mathsf{BDO{:}Y})))\end{aligned} \tag{5.10}$$

The negative containment pattern is similar. It has the form:

W of type X does not contain Z of type Y

The DL formalization differs according to the type W ($\mathsf{BPMNO{:}process}$/$\mathsf{BPMNO{:}embedded\ sub\text{-}process}$, $\mathsf{BPMNO{:}pool}$ or $\mathsf{BPMNO{:}business\_process\_diagram}$):

- $$\begin{aligned}&\mathsf{BDO{:}X} \sqcap \mathsf{BPMNO{:}embedded\_sub\_process/BPMNO{:}process} \sqsubseteq \\ &\quad \forall \mathsf{BPMNO{:}has\_embedded/BPMNO{:}has\_graphicals.}\neg(\mathsf{BPMNO{:}Z} \sqcap \mathsf{BDO{:}Y})\end{aligned} \tag{5.11}$$

- $$\begin{aligned}&\mathsf{BDO{:}X} \sqcap \mathsf{BPMNO{:}pool} \sqsubseteq \\ &\quad \forall \mathsf{BPMNO{:}has\_process\_ref.}(\mathsf{BPMNO{:}process} \sqcap \\ &\qquad \forall \mathsf{BPMNO{:}has\_graphicals.}\neg(\mathsf{BPMNO{:}Z} \sqcap \mathsf{BDO{:}Y}))\end{aligned} \tag{5.12}$$

- $$\begin{aligned}&\mathsf{BDO{:}X} \sqcap \mathsf{BPMNO{:}business\_process\_diagram} \sqsubseteq \\ &\quad \forall \mathsf{BPMNO{:}has\_diagram\_pools.}(\mathsf{BPMNO{:}pool} \sqcap \\ &\qquad \forall \mathsf{BPMNO{:}has\_process\_ref.}(\mathsf{BPMNO{:}process} \sqcap \\ &\qquad\quad \forall \mathsf{BPMNO{:}has\_graphicals.}\neg(\mathsf{BPMNO{:}Z} \sqcap \mathsf{BDO{:}Y})))\end{aligned} \tag{5.13}$$

Requirement (d) can also be seen as a containment constraint, though it is slightly more complex. In order to formalize the constraint in a readable manner we split it in two parts. We first define a concept goodAuthProcess which describes the second part of the constraint, that is, an authentication sub-process which contains both a to_log_in activity and a to_provide_customer_data activity[5].

$$\begin{aligned}
&\mathsf{goodAuthProcess} \equiv \\
&\quad \mathsf{BPMNO{:}embedded\_sub\_process} \sqcap \mathsf{BDO{:}to\_authenticate} \sqcap \\
&\quad \exists \mathsf{BPMNO{:}has\_embedded.}(\mathsf{BPMNO{:}activity} \sqcap \mathsf{BDO{:}to\_log\_in}) \sqcap \\
&\quad \exists \mathsf{BPMNO{:}has\_embedded.}(\mathsf{BPMNO{:}activity} \sqcap \mathsf{BDO{:}to\_provide\_customer\_data})
\end{aligned}$$

Then, we can require that all Customer pools are associated to a process that contains a goodAuthProcess:

$$\begin{aligned}
\mathsf{BPMNO{:}pool} \sqcap \mathsf{BDO{:}customer} \sqsubseteq &\exists \mathsf{BPMNO{:}has\_process\_ref.}(\mathsf{BPMNO{:}process} \sqcap \\
&\qquad \mathsf{BPMNO{:}has\_graphicals.goodAuthProcess})
\end{aligned}$$

Existence Constraints

Existence constraints are constraints of the form *exists X* or *non-exists X* and are used to represent the fact that a certain element X is present / absent in the BPD. In this perspective, hence, their form can be rephrased into *diagram X contains Y* and *diagram X does not contain Y*, thus falling in a particular case of containment constraints. The constraint may concern a plain BPMN element or a semantically annotated one. Two simple examples of existence constraint in the on-line shopping process in Figure 5.1 are provided by requirements (h) and (i).

A formal encoding of the existence constraint (h), hence, can be provided by simply asserting that the business process diagram encoded in the current BPKB must contain at least a pool object annotated with the BDO concept customer and a pool object annotated with on-line_shop:

$$\begin{aligned}
&\mathsf{BPMNO{:}business\_process\_diagram} \sqsubseteq \\
&\quad \exists \mathsf{BPMNO{:}has\_diagram\_pools.BDO{:}customer} \sqcap \\
&\quad \exists \mathsf{BPMNO{:}has\_diagram\_pools.BDO{:}on\text{-}line\_shop}
\end{aligned}$$

Similarly, requirement (i) can be realized by asserting that the processes referenced by the business process diagram encoded in the current BPKB must contain pools referencing

[5]We do not require them to be different activities.

processes containing no inclusive gateway:

$$\begin{aligned}&\mathsf{BPMNO{:}business\_process\_diagram} \sqsubseteq\\ &\quad \forall \mathsf{BPMNO{:}has\_diagram\_pools.(BPMNO{:}pool} \sqcap\\ &\qquad \forall \mathsf{BPMNO{:}has\_process\_ref.(BPMNO{:}process} \sqcap\\ &\qquad\quad \forall \mathsf{BPMNO{:}has\_graphicals.} \neg \mathsf{BPMNO{:}inclusive\_gateway))}\end{aligned}$$

By assuming that the diagram is formally correct with respect to the BPMNO formalization, the same merging axiom can be more simply encoded by asserting that processes in BPKB do not contain inclusive gateways:

$$\begin{aligned}&\mathsf{BPMNO{:}process} \sqsubseteq\\ &\quad \forall \mathsf{BPMNO{:}has\_graphicals.} \neg \mathsf{BPMNO{:}inclusive\_gateway}\end{aligned}$$

Abstracting the specific BDO types, hence, the pair of requirement patterns:

There exists Z of type Y

and

There does not exist Z of type Y

can be formalized in DL, similarly to the general containment constraints, according to the BPMNO-type Z of the object whose existence we are interested in verifying :

- In case Z is a pool, existence and non-existence constraints can be respectively encoded as:

$$\begin{aligned}&\mathsf{BPMNO{:}business\_process\_diagram} \sqsubseteq\\ &\quad \forall \mathsf{BPMNO{:}has\_diagram\_pools.(BPMNO{:}pool} \sqcap \mathsf{BDO{:}Y)}\end{aligned} \tag{5.14}$$

$$\begin{aligned}&\mathsf{BPMNO{:}business\_process\_diagram} \sqsubseteq\\ &\quad \exists \mathsf{BPMNO{:}has\_diagram\_pools.} \neg \mathsf{(BPMNO{:}pool} \sqcap \mathsf{BDO{:}Y)}\end{aligned} \tag{5.15}$$

- In case Z is a graphical object contained in a pool, instead, the DL formalization of the two constraints is the following:

$$\begin{aligned}&\mathsf{BPMNO{:}business\_process\_diagram} \sqsubseteq\\ &\quad \forall \mathsf{BPMNO{:}has\_diagram\_pools.(BPMNO{:}pool} \sqcap\\ &\qquad \forall \mathsf{BPMNO{:}has\_process\_ref.(BPMNO{:}process} \sqcap\\ &\qquad\quad \forall \mathsf{BPMNO{:}has\_graphicals.(BPMNO{:}Z} \sqcap \mathsf{BDO{:}Y)))}\end{aligned} \tag{5.16}$$

$$\begin{aligned}&\mathsf{BPMNO{:}business\_process\_diagram} \sqsubseteq \\ &\quad \exists \mathsf{BPMNO{:}has\_diagram\_pools}.(\mathsf{BPMNO{:}pool} \sqcap \\ &\qquad \exists \mathsf{BPMNO{:}has\_process\_ref}.(\mathsf{BPMNO{:}process} \sqcap \\ &\qquad\quad \exists \mathsf{BPMNO{:}has\_graphicals}.\neg(\mathsf{BPMNO{:}Z} \sqcap \mathsf{BDO{:}Y})))\end{aligned} \tag{5.17}$$

These patterns, of course, can be adapted to cases in which the BDO-type Y is not specified and the only BPMNO-type is available, or vice versa.

Enumeration Constraints

Enumeration constraints further refine containment constraints by stating that X *contains (at least / at most / exactly) n objects of type X*. A simple example of enumeration constraint which concerns a plain BPMN element is provided by requirements (j). An enumeration constraint which also involves semantically annotated objects is provided by requirement (k). Enumeration constraints can be encoded in Description Logics using the constructs: *number restriction* and *qualified number restriction* [12]. Number restrictions are written as $\geq nR$ (at-least restriction) and $\leq nR$ (at-most restriction), with n positive integer, while qualified number restrictions are written as $\geq nR.C$ and $\leq nR.C$. The difference between the two is that number restriction allows to write expressions such as, e.g., $(\leq 3)hasChild$, which characterise the set of individuals who have at most 3 children, while qualified number restriction allows to write expressions such as, e.g., $(\leq 3)hasChild.Female$, which characterise the set of individuals who have at most 3 female children. At-least and at-most operators can be combined to obtain statements of the form $=nR$.

A formalization of requirements (j) can then be provided by the DL statement:

$$\mathsf{BPMNO{:}gateway} \sqsubseteq (\leq 2)\mathsf{BPMNO{:}has\_gateway\_gate}$$

while a formalization of requirement (k) is given by:

$$\begin{aligned}\mathsf{BPMNO{:}pool} \sqsubseteq &\forall \mathsf{BPMNO{:}has\_process\_ref}. \\ &\qquad (=1)\mathsf{BPMNO{:}has\_graphicals}.\mathsf{BDO{:}to\_authenticate}\end{aligned}$$

Abstracting with respect to the specific BDO-type and BPMNO-type, we obtain constraints of the type:

W of BDO-type X contains (at least/at most/exactly) n objects Z of BDO-type Y

Their formalization is hence the same of containment constraints in which the subsumption relationship is constrained by $(\leq n/\geq n/= n)$. For example, in case W is an embedded sub-process, the DL formalization is:

$$\begin{aligned}&\mathsf{BDO{:}X} \sqcap \mathsf{BPMNO{:}embedded\_sub\_process} \sqsubseteq \\ &\quad (\leq n/\geq n/= n)\exists \mathsf{BPMNO{:}has\_embedded}.(\mathsf{BPMNO{:}Z} \sqcap \mathsf{BDO{:}Y})\end{aligned} \tag{5.18}$$

Precedence Constraints

Precedence constraints are used to represent the fact that certain graphical objects appear / do not appear before others in the BPD. They can be of several forms. Significant examples are: *X is always preceded by Y* in all possible paths made of sequence flows and *X is once preceded by Y* in at least a path composition of sequence flows and the corresponding negated forms: *X is not always preceded by Y* and *X is never preceded by Y*. Particular cases of these constraints are *X is always immediately preceded by Y*, *X is once immediately preceded by Y*, *X is not always immediately preceded by Y* and *X is never preceded by Y*. These constraints also require that X is a graphical object (not) immediately preceded by Y by means of a sequence flow. Finally the precedence constraint *X is activated by Y* requires that X is activated by Y by means of a message flow. Two simple examples of precedence constraint are provided by requirements (b) and (c).

Precedence constraints can be encoded in DL using specific BPMNO roles which formalize the connection between graphical objects. In particular the key roles we can use are:

- has_sequence_flow_source_ref and has_sequence_flow_target_ref.
- has_message_flow_source_ref and has_message_flow_target_ref.

These roles represent the SourceRef and TargetRef attributes of BPMN and identify which graphical elements the connecting object is connected from and to respectively. The first two roles (hereafter abbreviated with has_sf_source and has_sf_target for the sake of readability) refer to sequence flow connecting objects, while the other two roles (hereafter abbreviated with has_m_source and has_m_target for the sake of readability) refer to message flows.

Constraint (b) can be formalized in DL by means of two statements:

$$\begin{aligned}&\mathsf{BDO{:}to\_provide\_sensible\_data} \sqcap \mathsf{BPMNO{:}activity} \sqsubseteq \\ &\quad \forall \mathsf{BPMNO{:}has\_sf\_target}^{-}.\forall \mathsf{BPMNO{:}has\_sf\_source}.\mathsf{BDO{:}to\_read\_policy}^{*}\end{aligned} \tag{5.19}$$

$$\begin{aligned}&\mathsf{BDO{:}to\_read\_policy}^{*} \equiv \\ &\quad \neg \mathsf{BPMNO{:}start\_event} \sqcap \\ &\quad ((\mathsf{BDO{:}to\_read\_policy} \sqcap \mathsf{BPMNO{:}activity}) \sqcup \\ &\qquad \forall \mathsf{BPMNO{:}has\_sf\_target}^{-}.\forall \mathsf{BPMNO{:}has\_sf\_source}.\mathsf{BDO{:}to\_read\_policy}^{*})\end{aligned} \tag{5.20}$$

The statements above use has_sf_source and has_sf_target, together with an auxiliary concept BDO:to_read_policy*. In a nutshell the idea is that the concept BDO:to_provide_sensible_data is immediately preceded, in all paths defined by a sequence flow, by a graphical object of type BDO:to_read_policy*. This new concept is, in turn, defined as a graphical object which is not the start event and either it is an activity of type BDO:to_read_policy or it is preceded in all paths by BDO:to_read_policy*. By replacing to_provide_sensible_data, to_read_policy, and to_read_policy* with X, Y and Y^{*} in (5.19) and (5.20) and activity in (5.19) with W and activity in (5.20) with Z, we can obtain a general encoding of constraints of the form:

W of BDO-type X is always preceded by Z of BDO-type Y

Its formalization in DL is:

$$\begin{aligned}\mathsf{BDO{:}X} \sqcap \mathsf{BPMNO{:}W} \sqsubseteq &\forall \mathsf{BPMNO{:}has\_sf\_target}^{-}. \\ &\quad \forall \mathsf{BPMNO{:}has\_sf\_source}.\mathsf{BDO{:}Y}^{*} \\ \mathsf{BDO{:}Y}^{*} \equiv &\neg \mathsf{BPMNO{:}start\_event} \sqcap \\ &((\mathsf{BDO{:}Y} \sqcap \mathsf{BPMNO{:}Z}) \sqcup \\ &\quad \forall \mathsf{BPMNO{:}has\_sf\_target}^{-}.\forall \mathsf{BPMNO{:}has\_sf\_source}.\mathsf{BDO{:}Y}^{*})\end{aligned} \tag{5.21}$$

In addition, by replacing $\forall$ with $\exists$ we can obtain an encoding of:

W of BDO-type X is once preceded by Z of BDO-type Y

$$\begin{aligned}\mathsf{BDO{:}X} \sqcap \mathsf{BPMNO{:}W} \sqsubseteq & \exists \mathsf{BPMNO{:}has\_sf\_target}^{-}. \\ & \exists \mathsf{BPMNO{:}has\_sf\_source}.\mathsf{BDO{:}Y}^{*} \\ \mathsf{BDO{:}Y}^{*} \equiv & \neg \mathsf{BPMNO{:}start\_event} \sqcap \\ & ((\mathsf{BDO{:}Y} \sqcap \mathsf{BPMNO{:}Z}) \sqcup \\ & \exists \mathsf{BPMNO{:}has\_sf\_target}^{-}.\exists \mathsf{BPMNO{:}has\_sf\_source}.\mathsf{BDO{:}Y}^{*})\end{aligned} \tag{5.22}$$

Similarly, for the corresponding negated forms, i.e.,:

W of BDO-type X is not always preceded by Z of BDO-type Y

and

W of BDO-type X is never preceded by Z of BDO-type Y

the DL formalizations are:

$$\begin{aligned}\mathsf{BDO{:}X} \sqcap \mathsf{BPMNO{:}W} \sqsubseteq & \exists \mathsf{BPMNO{:}has\_sf\_target}^{-}. \\ & \exists \mathsf{BPMNO{:}has\_sf\_source}.\neg \mathsf{BDO{:}Y}^{*} \\ \mathsf{BDO{:}Y}^{*} \equiv & \neg \mathsf{BPMNO{:}start\_event} \sqcap \\ & ((\mathsf{BDO{:}Y} \sqcap \mathsf{BPMNO{:}Z}) \sqcup \\ & \forall \mathsf{BPMNO{:}has\_sf\_target}^{-}.\forall \mathsf{BPMNO{:}has\_sf\_source}.\mathsf{BDO{:}Y}^{*})\end{aligned} \tag{5.23}$$

and

$$\begin{aligned}\mathsf{BDO{:}X} \sqcap \mathsf{BPMNO{:}W} \sqsubseteq & \forall \mathsf{BPMNO{:}has\_sf\_target}^{-}. \\ & \forall \mathsf{BPMNO{:}has\_sf\_source}.\neg \mathsf{BDO{:}Y}^{*} \\ \mathsf{BDO{:}Y}^{*} \equiv & \neg \mathsf{BPMNO{:}start\_event} \sqcap \\ & ((\mathsf{BDO{:}Y} \sqcap \mathsf{BPMNO{:}Z}) \sqcup \\ & \exists \mathsf{BPMNO{:}has\_sf\_target}^{-}.\exists \mathsf{BPMNO{:}has\_sf\_source}.\mathsf{BDO{:}Y}^{*})\end{aligned} \tag{5.24}$$

If we replace the constraint (b) with a simpler constraint of the form *"the activity of providing personal data is once immediately preceded by an activity of reading the policy of the organization"* then, we do not need to introduce the auxiliary concept Y^* and the encoding is directly provided by the statement:

$$\begin{aligned}& \mathsf{BDO{:}to\_provide\_sensible\_data} \sqcap \mathsf{BPMNO{:}activty} \sqsubseteq \\ & \quad \exists \mathsf{BPMNO{:}has\_sf\_target}^{-}. \\ & \quad\quad \exists \mathsf{BPMNO{:}has\_sf\_source}.(\mathsf{BDO{:}to\_read\_policy} \sqcap \mathsf{BPMNO{:}activity})\end{aligned} \tag{5.25}$$

Again, we can abstract from the specific types and obtain a pattern of the form:

W of BDO-type X is always immediately preceded by Z of BDO-type Y

and

W of BDO-type X is once immediately preceded by Z of BDO-type Y

Their formalizations in DL are, respectively:

$$\begin{aligned}&\mathsf{BDO{:}X} \sqcap \mathsf{BPMNO{:}W} \sqsubseteq \\ &\quad \forall \mathsf{BPMNO{:}has\_sf\_target}^{-}. \\ &\qquad \forall \mathsf{BPMNO{:}has\_sf\_source}.(\mathsf{BDO{:}Y} \sqcap \mathsf{BPMNO{:}Z})\end{aligned} \tag{5.26}$$

and

$$\begin{aligned}&\mathsf{BDO{:}X} \sqcap \mathsf{BPMNO{:}W} \sqsubseteq \\ &\quad \exists \mathsf{BPMNO{:}has\_sf\_target}^{-}. \\ &\qquad \exists \mathsf{BPMNO{:}has\_sf\_source}.(\mathsf{BDO{:}Y} \sqcap \mathsf{BPMNO{:}Z})\end{aligned} \tag{5.27}$$

Their negated forms, instead, are:

W of BDO-type X is not always immediately preceded by Z of BDO-type Y

and

W of BDO-type X is never immediately preceded by Z of BDO-type Y

Their formalizations, hence, are:

$$\begin{aligned}&\mathsf{BDO{:}X} \sqcap \mathsf{BPMNO{:}W} \sqsubseteq \\ &\quad \exists \mathsf{BPMNO{:}has\_sf\_target}^{-}. \\ &\qquad \exists \forall \mathsf{BPMNO{:}has\_sf\_source}.\neg(\mathsf{BDO{:}Y} \sqcap \mathsf{BPMNO{:}Z})\end{aligned} \tag{5.28}$$

$$\begin{aligned}&\mathsf{BDO{:}X} \sqcap \mathsf{BPMNO{:}W} \sqsubseteq \\ &\quad \forall \mathsf{BPMNO{:}has\_sf\_target}^{-}. \\ &\qquad \forall \mathsf{BPMNO{:}has\_sf\_source}.\neg(\mathsf{BDO{:}Y} \sqcap \mathsf{BPMNO{:}Z})\end{aligned} \tag{5.29}$$

In general, however, it is also possible to define patterns of the form:

The activity of BDO-type X is once immediately preceded allowing gateways in-between by the activity of BDO-type Y by using the expression *once immediately preceded allowing gateways in-between* for indicating that activity X immediately precedes activity Y or that there exists a path between the two activities that traverses only gateways. The formalization of this pattern is:

$$\begin{aligned}\mathsf{BDO{:}X} \sqcap \mathsf{BPMNO{:}W} \sqsubseteq &\exists \mathsf{BPMNO{:}has\_sf\_target}^{-}.\\ &\quad\exists \mathsf{BPMNO{:}has\_sf\_source.BDO{:}Y}^{*}\\ \mathsf{BDO{:}Y}^{*} \equiv &\mathsf{BPMNO{:}gateway} \sqcap\\ &((\mathsf{BDO{:}Y} \sqcap \mathsf{BPMNO{:}activity}) \sqcup\\ &\quad\forall \mathsf{BPMNO{:}has\_sf\_target}^{-}.\forall \mathsf{BPMNO{:}has\_sf\_source.BDO{:}Y}^{*})\end{aligned} \tag{5.30}$$

DL translation for the *always immediately preceded allowing gateways in-between* (indicating that there exist one or more paths between activity X and activity Y and that each of these paths either does not traverse other process elements or traverses only gateways) is analogous.

Finally, patterns of the form *X is always followed by Y*, *X is once followed by Y*, *X is always immediately followed by Y*, *X is once immediately followed by Y* and their negated forms can be encoded by swapping has_sf_source and has_sf_target roles. For example, the generic pattern:

W of BDO-type X is once immediately followed by Z of BDO-type Y

is encoded as:

$$\begin{aligned}&\mathsf{BDO{:}X} \sqcap \mathsf{BPMNO{:}W} \sqsubseteq\\ &\quad\exists \mathsf{BPMNO{:}has\_sf\_source}^{-}.\\ &\qquad\exists \mathsf{BPMNO{:}has\_sf\_target.}(\mathsf{BDO{:}Y} \sqcap \mathsf{BPMNO{:}Z})\end{aligned} \tag{5.31}$$

In the last group of these constraints we find those involving the messaging flow. To formalise (c) we need to check that the activities annotated with BDO:to_read_policy are activated by an intermediate event (message) which refers to a message flow originated by an activity of BDO-type BDO:to_provide_policy_data:

$$\begin{aligned}&\mathsf{BDO{:}to\_read\_policy} \sqcap \mathsf{BPMNO{:}activity} \sqsubseteq\\ &\quad\exists \mathsf{BPMNO{:}has\_sf\_target}^{-}.\exists \mathsf{BPMNO{:}has\_sf\_source.}\\ &\qquad(\mathsf{BPMNO{:}intermediate\_event} \sqcap\\ &\qquad\quad\exists \mathsf{BPMNO{:}has\_m\_target}^{-}.\exists \mathsf{BPMNO{:}has\_m\_source.}\\ &\qquad\qquad(\mathsf{BDO{:}to\_provide\_policy\_data} \sqcap \mathsf{BPMNO{:}activity}))\end{aligned} \tag{5.32}$$

Again, by replacing the specific BDO concepts with X and Y, and the specific BPMNO concepts with W and Z, we can obtain a schematic encoding of constraints of the form:

W of BDO-type X is activated by Z of BDO-type Y

that is:

$$\begin{aligned}\mathsf{BDO{:}X} \sqcap \mathsf{BPMNO{:}W} \sqsubseteq\, & \exists \mathsf{BPMNO{:}has\_sf\_target}^{-}.\exists \mathsf{BPMNO{:}has\_sf\_source}.\\ & (\mathsf{BPMNO{:}intermediate\_event} \sqcap \\ & \quad \exists \mathsf{BPMNO{:}has\_m\_target}^{-}.\exists \mathsf{BPMNO{:}has\_m\_source}.\\ & \quad (\mathsf{BDO{:}Y} \sqcap \mathsf{BPMNO{:}Z}))\end{aligned} \tag{5.33}$$

Similarly, for its negative form:

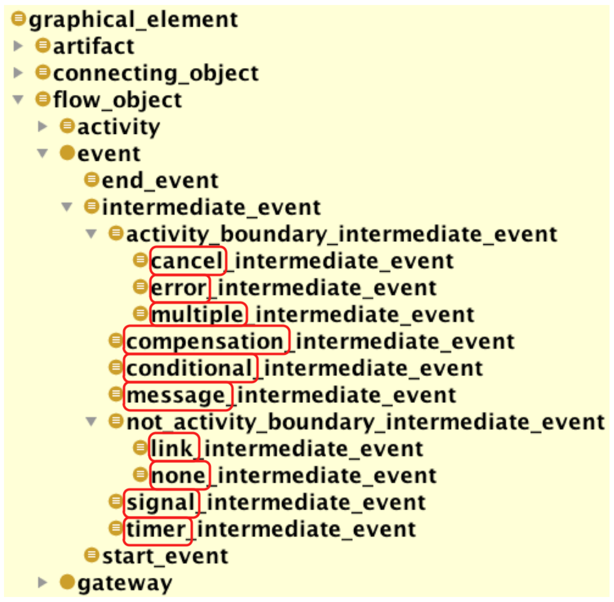
W of BDO-type X is not activated by Z of BDO-type Y

$$\begin{aligned}\mathsf{BDO{:}X} \sqcap \mathsf{BPMNO{:}W} \sqsubseteq\, & \forall \mathsf{BPMNO{:}has\_sf\_target}^{-}.\forall \mathsf{BPMNO{:}has\_sf\_source}.\\ & (\mathsf{BPMNO{:}intermediate\_event} \sqcap \\ & \quad \forall \mathsf{BPMNO{:}has\_m\_target}^{-}.\forall \mathsf{BPMNO{:}has\_m\_source}.\neg\\ & \quad (\mathsf{BDO{:}Y} \sqcap \mathsf{BPMNO{:}Z}))\end{aligned} \tag{5.34}$$

Exception Handling Constraints

These constraints are expressions used to represent the way specific exceptions should be handled. They describe structural properties of BPMN diagrams and, similarly to the other types of structural constraints, they can have many different forms. They can specify simple requirements stating the need for an exception handling mechanism at a certain point in the process, as in the requirement (e), or more complex issues, as in the requirement (g).

The formalization of exception handling constraints is based on the representation of BPMN Intermediate Events (None, Message, Timer, Error, Cancel, Compensation, Conditional, Link, Signal, and Multiple) in BPMNO. Intermediate events belong to the extended set of BPMN Graphical Elements described in [131] and are the mechanism BPMN suggests to use to represent exception or compensation handling. Intermediate events are part of the graphical objects of BPMNO, as depicted in Figure 5.3. In accordance with the properties of intermediate events [131] encoded in BPMNO, these elements are further classified, via reasoning, in different groups: the events classified as activity_boundary_ intermediate_events can only appear on the boundary of an activity (i.e, on the boundary of its



```
graphical_element
  artifact
  connecting_object
  flow_object
    activity
    event
      end_event
      intermediate_event
        activity_boundary_intermediate_event
          cancel_intermediate_event
          error_intermediate_event
          multiple_intermediate_event
        compensation_intermediate_event
        conditional_intermediate_event
        message_intermediate_event
        not_activity_boundary_intermediate_event
          link_intermediate_event
          none_intermediate_event
        signal_intermediate_event
        timer_intermediate_event
      start_event
    gateway
```

Figure 5.3: The classification of intermediate events

graphical representation), the events classified as not_activity_boundary_intermediate_events must not appear on the boundary of an activity, and finally the events which are not classified under any of these concepts can appear in both circumstances. Two BPMNO roles are used to describe the attributes of intermediate events:

- has_intermediate_event_target, which encodes the Target attribute. This attribute is used to describe the fact that the intermediate event is attached to the boundary of an activity.

- has_intermediate_event_trigger encodes the Trigger attribute. This attribute is used to describe the type of trigger expected for an intermediate event.

For the sake of readability, we abbreviate the two roles above with has_target and has_trigger, respectively.

The constraint corresponding to requirement (e) can be formalized in DL by means of the following statement where, for the sake of presentation, we separately define the reserveProductOn-line concept used to denote the activities contained in the On-line Shop pool and annotated with the to_reserve_product concept:

$$\begin{aligned}&\mathsf{reserveProductOn\text{-}line} \equiv\\&\quad \mathsf{BPMNO{:}activity} \sqcap\\&\quad \mathsf{BDO{:}to\_reserve\_product} \sqcap \exists \mathsf{has\_graphicals}^{-}.\\&\qquad \exists \mathsf{has\_process\_ref}^{-}.\mathsf{BDO{:}on\text{-}line\_shop}\end{aligned} \tag{5.35}$$

$$\begin{aligned}&\mathsf{reserveProductOn\text{-}line} \sqsubseteq \exists \mathsf{BPMNO{:}has\_target}^{-}.\\&\quad (\mathsf{BPMNO{:}error\_intermediate\_event} \sqcap \mathsf{BDO{:}product\_unavailability})\end{aligned} \tag{5.36}$$

Requirement (f) can instead be formalized in DL by a statement which makes use of precedence constraints. In the following we use $\forall\mathsf{BPMNO{:}is\_followed\_by}$ as a shorthand for $\forall\mathsf{BPMNO{:}has\_sf\_target}^{-}.\ \forall\mathsf{BPMNO{:}has\_sf\_source}$. Similarly, for $\exists\mathsf{BPMNO{:}is\_followed\_by}$.

$$\begin{aligned}\mathsf{productUnavailabilityInOn\text{-}line} \equiv &(\mathsf{BPMNO{:}error\_intermediate\_event} \sqcap\\&\mathsf{BDO{:}product\_unavailability} \sqcap\\&\forall \mathsf{BPMNO{:}has\_target.reserveProductOn\text{-}line})\end{aligned} \tag{5.37}$$

$$\begin{aligned}\mathsf{productUnavailabilityInOn\text{-}line} \sqsubseteq &\forall \mathsf{BPMNO{:}is\_followed\_by}.\\&\quad (\mathsf{BPMNO{:}parallel\_gateway} \sqcap\\&\quad \exists \mathsf{BPMNO{:}is\_followed\_by}.\\&\qquad (\mathsf{BPMNO{:}task} \sqcap \mathsf{BDO{:}to\_warn\_buyer}) \sqcap\\&\quad \exists \mathsf{BPMNO{:}is\_followed\_by}.\\&\qquad (\mathsf{BPMNO{:}sub\_process} \sqcap \mathsf{BDO{:}to\_order\_product}))\end{aligned} \tag{5.38}$$

Axiom (5.37) defines the class product_unavailability error events caught by the activity of reserving products in the On-line Shop pool, while axiom (5.38) states that error events of this BDO-type must be followed by a parallel gateway which leads to two activities: a task for warning the buyer, and a sub-process which takes care of ordering the missing products.

Finally, in order to ease the readability of the formalization of requirement (g), we split it into three parts. The first one asserts that the activity sending customer data contained in a sub-process for the log-in has to be followed by an error event originated from the On-line Shop pool and representing the failure of a compulsory log-in (*The activity of sending customer data in the log-in sub-process is always immediately followed by a compulsory log-in error event originated in the On-line Shop pool*). The DL formalization of such a requirement is the following:

$$\begin{aligned}\mathsf{sendCustomerDataLogin} \equiv{} & \mathsf{BPMNO{:}activity} \sqcap \\ & \mathsf{BDO{:}to\_send\_customer\_data} \sqcap \\ & \exists \mathsf{BPMNO{:}has\_embedded}^{-}. \\ & \quad (\mathsf{BPMNO{:}embedded\_sub\_process} \sqcap \mathsf{BDO{:}to\_log\_in})\end{aligned} \tag{5.39}$$

$$\begin{aligned}\mathsf{graphicalElementOnline} \equiv{} & \mathsf{BPMNO{:}graphical\_element} \sqcap \\ & \exists \mathsf{BPMNO{:}has\_graphicals}^{-}.\exists \mathsf{BPMNO{:}has\_process\_ref}^{-} \\ & \quad (\mathsf{BPMNO{:}pool} \sqcap \mathsf{BDO{:}on\text{-}line\_shop})\end{aligned} \tag{5.40}$$

$$\begin{aligned}\mathsf{loginFailureFromOnline} \equiv{} & \mathsf{BPMNO{:}error\_intermediate\_event} \sqcap \\ & \mathsf{BDO{:}compulsory\_log\_in\_failure} \sqcap \\ & \exists \mathsf{BPMNO{:}has\_m\_target}^{-}. \\ & \quad \exists \mathsf{BPMNO{:}has\_m\_source}.\mathsf{graphicalElementOnline}\end{aligned} \tag{5.41}$$

$$\mathsf{sendCustomerDataLogin} \sqsubseteq \forall \mathsf{BPMNO{:}is\_followed\_by}.\mathsf{loginFailureFromOnline} \tag{5.42}$$

Axiom (5.39) defines the class of to_send_customer_data activities contained in a to_log_in sub-process, axiom (5.40) defines the class of graphical elements contained in the On-line Shop pool, axiom (5.41) the class of compulsory_log_in_failure error intermediate events originated by a graphicalElementOnline and, finally, axiom (5.42) states that the sendCustomerDataLogin activities have to be followed by a compulsory_log_in_failure event.

The second part of the requirement (g), states, instead, that *the log-in sub-process has to catch a compulsory log-in error event.* Its DL formalization is the following:

$$\begin{aligned}& \mathsf{BDO{:}to\_log\_in} \sqcap \mathsf{BPMNO{:}sub\_process} \sqsubseteq \\ & \quad \exists \mathsf{BPMNO{:}has\_target}^{-}. \\ & \qquad (\mathsf{BPMNO{:}error\_intermediate\_event} \sqcap \mathsf{BDO{:}compulsory\_log\_in\_failure})\end{aligned} \tag{5.43}$$

Finally, the third part is related to the exception handler: *the compulsory log-in failure error event caught by the login sub-process is always followed by an end event.* The corresponding DL statement is the following:

$$\begin{aligned}\mathsf{loginFailureLoginSubprocess} \equiv{}& \mathsf{BPMNO{:}error\_intermediate\_event} \sqcap \\ & \mathsf{BDO{:}compulsory\_log\_in\_failure} \sqcap \\ & \exists \mathsf{BPMNO{:}has\_target}^{-}. \\ & \quad (\mathsf{BPMNO{:}embedded\_sub\_process} \sqcap \mathsf{BDO{:}to\_log\_in})\end{aligned} \tag{5.44}$$

$$\begin{aligned}\mathsf{loginFailureLoginSubprocess} \sqsubseteq{}& \exists \mathsf{BPMNO{:}has\_sf\_source}^{-}. \\ & \exists \mathsf{BPMNO{:}has\_sf\_target}.\mathsf{BPMNO{:}error\_end\_event}\end{aligned} \tag{5.45}$$

In detail, axiom (5.44) defines the class of compulsory_log_in_failure error events caught by to_log_in sub-processes and axiom (5.45) states that it has been immediately followed by an error end event.

If we abstract away from the specific activities in requirement (e) we can obtain a pattern of the form:

The activity X has ***always to catch an error event*** *Y*

which can be encoded in an axiom skeleton of the form:

$$\begin{aligned}\mathsf{BDO{:}X} \sqcap \mathsf{BPMNO{:}activity} \sqsubseteq{}& \exists \mathsf{BPMNO{:}has\_target}^{-}. \\ & (\mathsf{BPMNO{:}error\_intermediate\_event} \sqcap \mathsf{BDO{:}Y})\end{aligned}$$

Abstracting also from the specific BPMNO-type Z of the event, we can obtain a pattern of the form:

The activity X has ***always to catch an event*** *Z of BDO-type Y*

which can be encoded in an axiom skeleton of the form:

$$\mathsf{BDO{:}X} \sqcap \mathsf{BPMNO{:}activity} \sqsubseteq \exists \mathsf{BPMNO{:}has\_target}^{-}.(\mathsf{BPMNO{:}Z} \sqcap \mathsf{BDO{:}Y}) \tag{5.46}$$

The definition of patterns that specify how exceptions are handled is more complex. This is due to the different specific ways in which an exception may be handled in a single process. Nevertheless, a number of papers focused on the description of error handling patterns exists (see e.g., [157]). These efforts can provide some guideline on how to define classes of constraints for typical exception handling patterns. Hereafter, we denote with SC(BPMNO,BDO) the set of axioms encoding structural constraints (including exception handling constraints).

Combining different constraints

By combining containment, enumeration precedence and exception handling constraints, we can encode more complex requirements. An example is provided by the following requirement:

All the paths that originate from the exclusive gateways contained in the On-line Shop pool must start with an event which comes from the Customer pool.

We first formalize the first sentence:

$$\begin{aligned}
&\mathsf{onLineShopPool} \equiv \mathsf{BPMNO{:}pool} \sqcap \mathsf{BDO{:}on\text{-}line\_shop}\\
&\mathsf{onLineShopPool} \sqsubseteq \exists \mathsf{BPMNO{:}has\_process\_ref.}(\mathsf{BPMN{:}process} \sqcap\\
&\quad \exists \mathsf{BPMNO{:}has\_graphicals.BPMNO{:}event\_based\_exclusive\_gateway})
\end{aligned}$$

To formalize the remaining part we introduce some additional concepts: the first two concepts represent graphical elements which reside in the Customer and On-line Shop pools respectively; the third for representing “split” exclusive event gateways, that is, exclusive event gateways which multiply paths; and the fourth for events which originate from the Customer pool. Finally, we define a concept for split event exclusive gateways in the On-line shop pool. Note that since BPMN does not distinguish between “split” and “merge” gateways, the only way to characterise the gateway where an exclusive path starts is to rely on some of its properties. We consider here a “merge” exclusive event gateway a gateway that has a single sequence flow exiting from it.[6]

[6] An alternative approach would be to “force” the designer to annotate gateways with a “split” or “merge” label but we do not follow this approach as this would mean adding semantics to the BPMN language itself.

$$
\begin{aligned}
\mathsf{customerPool} \equiv & \mathsf{BPMNO{:}pool} \sqcap \mathsf{BDO{:}customer}\\
\mathsf{customerGraphicalElement} \equiv & \mathsf{BPMNO{:}graphical\_element} \sqcap\\
& \exists \mathsf{has\_graphicals}^{-}.\exists \mathsf{has\_process\_ref}^{-}.\mathsf{customerPool}\\
\mathsf{onLineShopGraphicalElement} \equiv & \mathsf{BPMNO{:}graphical\_element} \sqcap\\
& \exists \mathsf{BPMNO{:}has\_graphicals}^{-}.\exists \mathsf{has\_process\_ref}^{-}.\mathsf{onLinePool}\\
\mathsf{splitEventExclusiveGateway} \equiv & \mathsf{BPMNO{:}event\_based\_exclusive\_gateway} \sqcap\\
& \exists (> 1)\mathsf{BPMNO{:}has\_sf\_source}^{-}.\top\\
\mathsf{eventFromCustomer} \equiv & \mathsf{BPMNO{:}intermediate\_event} \sqcap\\
& \exists.\mathsf{BPMNO{:}has\_m\_source}^{-}.\mathsf{customerGraphicalElement}\\
\mathsf{onLineShopSplitEventExclGateway} \equiv & \mathsf{splitEventExclusiveGateway} \sqcap\\
& \quad \mathsf{onLineShopGraphicalElement}\\
\mathsf{onLineShopSplitEventExclGateway} \sqsubseteq & \forall \mathsf{BPMNO{:}has\_sf\_source}^{-}.\\
& \quad \forall \mathsf{BPMNO{:}has\_sf\_target}.\mathsf{eventFromCustomer}
\end{aligned}
$$

Another example of requirement combining different types of constraints is the requirement (g) previously formalized in (5.39)–(5.42), (5.43) and (5.44)–(5.45).

Relaxed constraints

Due to expressiveness limitation imposed by Description Logics and by the fact that we want to remain in a decidable version of OWL, there are also constraints on the static parts of the BPMN diagram which are are not representable in our approach. In particular all the properties that, once translated into first order logic, require more than two variables cannot be represented. A typical example of this kind of constraint is the fact that X *mutually_excludes* Y, that is: X and Y are always preceded by the same exclusive gateway. In fact we can express this property as follows, where *precede* is used to indicate the precedence relation between graphical elements:

$$
\begin{gathered}
\forall X.\forall Y.\exists Z(xor(Z) \wedge precede(Z, X) \wedge precede(Z, Y) \wedge \forall W.(precede(Z, W) \wedge\\
precede(W, X) \wedge precede(W, Y)) \rightarrow \neg gateway(W))
\end{gathered}
$$

We can instead represent a weaker version of the constraint above, which states that X and Y are immediately preceded by an exclusive XOR gateway, by using precedence constraints. Similar limitations apply to constraints involving parallel gateways.

Table 5.2 provides a summary of the main (and most general) requirement patterns analysed and of their formalization in DL.

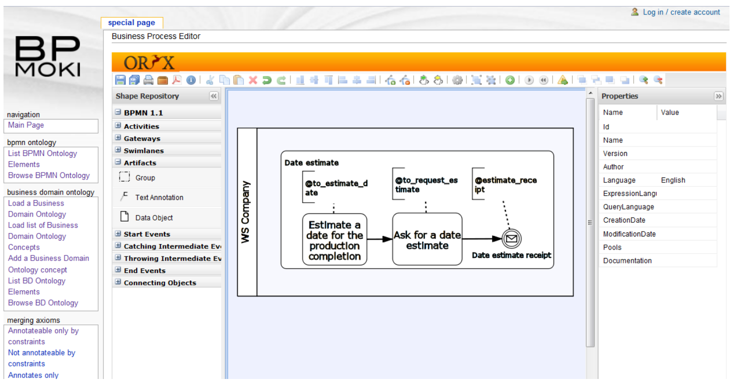
Category	Pattern	Formalization
Containment	*W of BDO-type X* ***contains*** *Z of BDO-type Y*	(5.8), (5.9) and (5.10)
Constraints	*W of BDO-type X* ***does not contain*** *Z of BDO-type Y*	(5.11), (5.12) and (5.13)
Existence	*It exists Z of BDO-type Y*	(5.14) and (5.16)
Constraints	*It does not exist Z of BDO-type Y*	(5.15) and (5.17)
Enumeration Constraints	*W of BDO-type contains (at least/at most/ exactly) n objects Z of BDO-type Y*	e.g., (5.18)
	W of BDO-type X is always preceded by Z of BDO-type Y	(5.21)
Precedence	*W of BDO-type X is once preceded by Z of BDO-type Y*	(5.22)
	W of BDO-type X is never preceded by Z of BDO-type Y	(5.24)
	W of BDO-type X is not always preceded by Z of BDO-type Y	(5.23)
	W of BDO-type X is always immediately preceded by Z of BDO-type Y	(5.26)
	W of BDO-type X is once immediately preceded by Z of BDO-type Y	(5.27)
	W of BDO-type X is not always immediately preceded by Z of BDO-type Y	(5.28)
	W of BDO-type X is never immediately preceded by Z of BDO-type Y	(5.29)
Constraints	*The activity of BDO-type X is once immediately preceded allowing gateways in-between by the activity of BDO-type Y*	(5.30)
	...	
	W of BDO-type X is immediately followed by Z of BDO-type Y	(5.31)
	...	
	W of BDO-type X is activated by Z of BDO-type Y	(5.33)
	W of BDO-type X is not activated by Z of BDO-type Y	(5.34)
Exception Handling Constraints	*The activity of BDO-type X has always to catch Z of BDO-type Y*	(5.46)

Table 5.2: Requirement Patterns

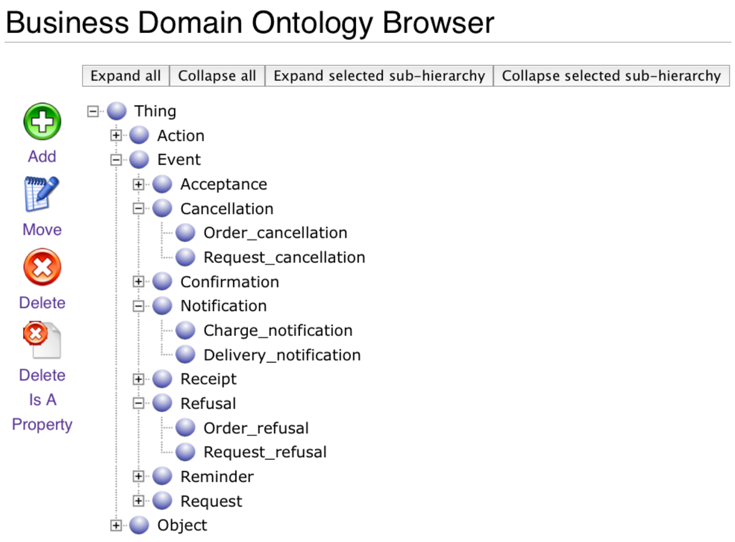
5.1.4 User-friendly Constraint Representation

Besides the classification of requirement patterns, some work has been done for providing a user-friendly support to business experts, for business process editing and semantic annotation, for ontology editing and for constraint definition. In detail, we realized BP-MoKi, a tool based on Semantic MediaWiki (SMW)[7] [95] and thus also supporting the collaborative aspect of semantically annotated processes. Its current functionalities allow to design processes (by means of the Oryx[8] process editor, a state of the art collaborative tool for the graphical modelling of business processes), to import, edit, manage and visualize ontologies, to define merging axioms, as well as, to check the correctness of the annotations (with respect to the merging axioms). For example, Figure 5.4 shows two views of the process and ontology editing functionalities, respectively. Though, in the current version of the tool the support to the definition of constraints is limited to the merging axioms (Figure 5.5 shows an example of template for the definition of an

[7]`http://semantic-mediawiki.org` - We currently use MediaWiki v1.14 and SMW v1.4.2.
[8]`http://bpt.hpi.uni-potsdam.de/Oryx`



(a) Process model editing in BP-MoKi



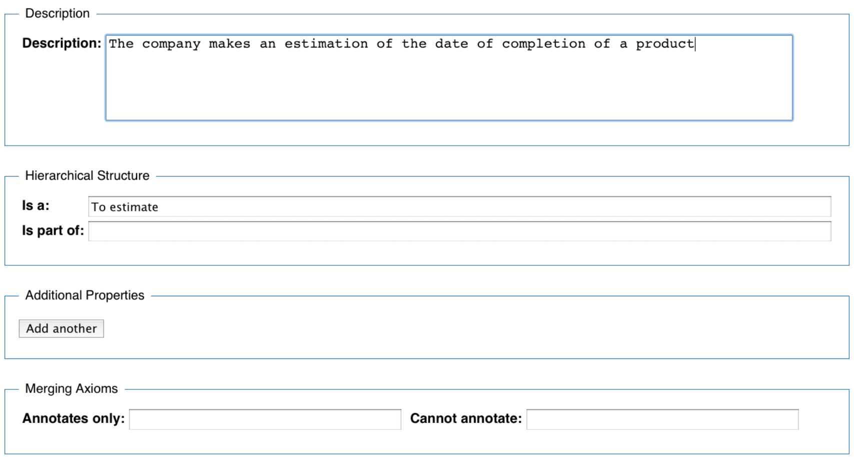
(b) Ontology editing in BP-MoKi

Figure 5.4: Two views of the functionalities provided by BP-MoKi

"annotates only" merging axiom of the BDO concept to_estimate_date), it can similarly, be extended to structural constraint patterns.

5.2 Constraint Verification

By encoding all the information about a semantically annotated business process into a logical knowledge base (as described in Chapter 4), several reasoning services over it can be implemented. Key reasoning services we present in this section are: **compatibility checking of process constraints** and **constraints verification** over an annotated BPD.


Description

Description: The company makes an estimation of the date of completion of a product

Hierarchical Structure

Is a: To estimate

Is part of:

Additional Properties

Add another

Merging Axioms

Annotates only: Cannot annotate:

Figure 5.5: User-friendly definition of an "annotates only" merging axiom (bottom box) of the BDO concept to_estimate_date

5.2.1 Compatibility Checking of Process Constraints

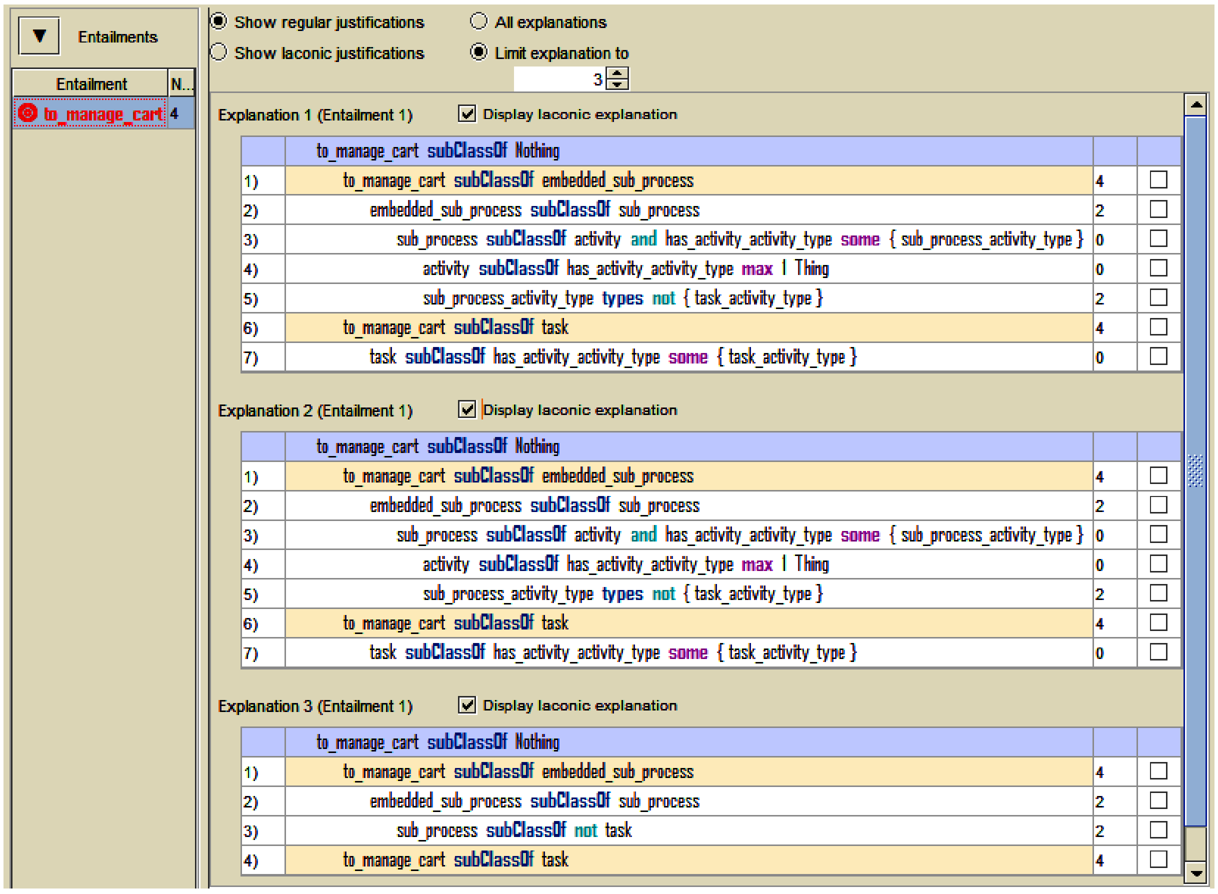
In formalizing the requirements that an annotated business process has to satisfy, the constraints specified by the user may generate inconsistencies in the resulting BPKB. This is due to the introduction of at least a process constraint which is incompatible with the axioms encoded in BPMNO or in BDO, or with other process constraints. The detection of incompatible process constraints can be automatically performed by verifying the consistency of the Tbox component of the BPKB:

$$\mathsf{BPMNO} \cup \mathsf{BDO} \cup \mathsf{MA}(\mathsf{BPMNO}, \mathsf{BDO}) \cup \mathsf{SC}(\mathsf{BPMNO},\mathsf{BDO})$$

with a standard state-of-the-art OWL DL reasoner. In the case of inconsistency, when some unsatisfiable class is detected, the usage of DL reasoners and explanation techniques similar to the ones described in [80] can be also useful to provide justifications to the business experts.

For example, Figure 5.6 shows an explanation (obtained by using the Explanation Workbench plugin for Protégé-4[9]) for the unsatisfiable concept to_manage_cart in case the assertion BDO:to_manage_cart $\sqsubseteq$BPMN:task is added to the knowledge base, together with constraint (5.4). The Explanation Workbench plugin finds four possible justifications for the unsatisfiable concept (only three are shown). For each justification, the set of axioms making the class unsatisfiable is reported. For instance, in the example, the first two justifications, which in their laconic form are equivalent, concern the conflict between

[9]http://owl.cs.manchester.ac.uk/explanation/


Entailments

Entailment	N...
to_manage_cart	4

[✓ Show regular justifications] Show laconic justifications

All explanations [✓ Limit explanation to] 3

Explanation 1 (Entailment 1) ✓ Display laconic explanation

	to_manage_cart subClassOf Nothing		
1)	to_manage_cart subClassOf embedded_sub_process	4	☐
2)	embedded_sub_process subClassOf sub_process	2	☐
3)	sub_process subClassOf activity and has_activity_activity_type some { sub_process_activity_type }	0	☐
4)	activity subClassOf has_activity_activity_type max 1 Thing	0	☐
5)	sub_process_activity_type types not { task_activity_type }	2	☐
6)	to_manage_cart subClassOf task	4	☐
7)	task subClassOf has_activity_activity_type some { task_activity_type }	0	☐

Explanation 2 (Entailment 1) ✓ Display laconic explanation

	to_manage_cart subClassOf Nothing		
1)	to_manage_cart subClassOf embedded_sub_process	4	☐
2)	embedded_sub_process subClassOf sub_process	2	☐
3)	sub_process subClassOf activity and has_activity_activity_type some { sub_process_activity_type }	0	☐
4)	activity subClassOf has_activity_activity_type max 1 Thing	0	☐
5)	sub_process_activity_type types not { task_activity_type }	2	☐
6)	to_manage_cart subClassOf task	4	☐
7)	task subClassOf has_activity_activity_type some { task_activity_type }	0	☐

Explanation 3 (Entailment 1) ✓ Display laconic explanation

	to_manage_cart subClassOf Nothing		
1)	to_manage_cart subClassOf embedded_sub_process	4	☐
2)	embedded_sub_process subClassOf sub_process	2	☐
3)	sub_process subClassOf not task	2	☐
4)	to_manage_cart subClassOf task	4	☐

Figure 5.6: Explanation generation

the sub-process and the task has_activity_type role. In fact, to_manage_cart is an embedded_sub_process, i.e., a sub_process that, in turn, is an activity, whose only activity_type role (an activity can have only one activity type) is a sub_process_activity_type that, however, cannot be a task_activity_type. On the other hand, to_manage is also a task and the has_activity_type of a task is a task_activity_type. The third explanation, instead, is simpler. It relies on the fact that to_manage, that is an embedded_sub_process and hence a sub_process, cannot be a task. However, to_manage is also a task, hence the class is unsatisfiable.

5.2.2 Constraints Verification over an Annotated BPD

Given an Abox A_β containing the OWL representation of a semantically annotated BPD β, the extension of the mechanism used for the compatibility checking to the constraint checking on annotated BPDs (i.e., the verification of the consistency of the knowledge base $\mathsf{BPMNO} \cup \mathsf{BDO} \cup \mathsf{MA(BPMNO, BDO)} \cup \mathsf{SC(BPMNO,BDO)} \cup A_\beta$) might seem straightforward. On the contrary, such a verification requires some care. This because the OWL semantics is based on the *Open World Assumption* (i.e., a failure in proving a statement does not imply that the statement is false) and does not satisfy the *Unique Names Assumption* (i.e., two entities with different identifiers are distinct objects), which makes it difficult to use OWL for data validation where complete knowledge can be assumed (i.e., a *closed world*), like in the case of an annotated BPD.

For example, OWL allows to encode the requirement (e), as shown in equation (5.35), but having an activity in the On-line Shop pool of BDO-type to_reserve_product with no error intermediate event of BDO-type product_unavailability attached to its boundary would not cause a logical inconsistency in the BPKB. In fact by reasoning in Open World Assumption, a failure of proving that any of the product_unavailability error intermediate events explicitly mentioned in the Abox A_β is attached to the boundary of an activity of BDO-type to_reserve_product in the On-line Shop pool would not imply that this element does not exist.

However, as discussed in [161], it is possible to define an *Integrity Constraint* (IC) semantics for OWL axioms in order to enable closed world constraints validation: constraints are written as standard OWL axioms but are interpreted with a different semantics for constraint validation. To support the validation of IC in OWL, the Pellet IC Validator[10], a prototype tool that extends the Pellet OWL reasoner by interpreting OWL axioms with IC semantics, can be used [161]. Technically, each axiom representing an IC

[10]http://clarkparsia.com/pellet

is first translated to a SPARQL query, and then executed by a SPARQL query engine over the Pellet reasoner to perform the validation over a given set of individuals[11]. By reasoning with IC semantics, the existence of an activity of BDO-type to_reserve_product in the On-line Shop pool, such that none of the error intermediate events of BDO-type product_unavailability defined in the BPKB is attached to its boundary, would cause a violation of requirement (e).

5.3 Constraint Checking Performance Evaluation

We performed a preliminary experiment in order to provide a first evaluation of the performance of semantic reasoning techniques used to support the verification of constraints over annotated BPDs. In particular, the goal of the evaluation was to provide an estimate of the performance when (i) checking the consistency of the BPKB, (ii) transforming an annotated BPD into an OWL Abox and (iii) validating the populated BPKB against constraints.

The evaluation study that we conducted comprised two experiments[12]. In the first experiment we considered six different processes (`P1`- `P6`) of increasing size (with a number of process graphical elements ranging from 92 to 475), and, for each of them, a single requirement (of the same kind of requirement (e)). `P1`, `P2`, `P4` and `P6` describe an on-line shopping process. In detail, `P1`, `P2` and `P6` are three incremental versions of the same process, where `P1` and `P2` only describe a part of the process (i.e., the product browsing and the cart management, leaving out the checkout phase), while `P6` represent the whole process. `P4`, instead is a variant of the complete process `P6`. Moreover, `P1` contains only the “Customer” pool, while `P2`, `P4` and `P6` contain both the “Customer” and the “On-line Shop” pool, thus describing the process from both the perspectives. `P3` describes the procedure about the management of a mortgage request performed by a potential customer (i.e., its acceptance or refusal by the mortgage company). This process also involves two pools, the “Potential Customer” and the “Mortgage Company” pool. Finally, `P5` describes a seller supplier chain for product management. In detail, in this process four pools are involved: the “Retail Seller Company”, the “Warehouse Company”, the “Delivery Company” and the actual “Transporter”. The product/service supplier chain is initiated by the Retail Seller Company after a warehouse check and evaluation. The purpose of this experiment was to study the performance of the semantic technology tools

[11]The Pellet reasoner provides also some basic automatic explanations of why an IC is violated.

[12]The machine used for both the experiments is a desktop PC with an Intel Core i7 2x2.80GHz processor, 6Gb of RAM, and running Linux Red Hat 5.

used, as the size of the BPKB (in terms of instances) grows. The main characteristics of the domain ontologies used to annotate the processes are reported in the top rows of Table 5.3. The DL expressiveness of the BPKBs considered is $\mathcal{ALCHOIN}(\mathcal{D})$[13].

		P1	**P2**	**P3**	**P4**	**P5**	**P6**
Process	*Graphical Process Elements*	92	175	237	327	387	475
Domain Ontology	*Classes*	124	124	101	114	79	124
	Class Axioms	133	133	101	113	77	133
Consistency Phase	*Consistency Check Time (s)*	1.422 (0.100)	1.447 (0.098)	1.439 (0.094)	1.445 (0.104)	1.422 (0.097)	1.429 (0.099)
Population Phase	*Added Individuals*	188	361	493	676	806	977
	Added Assertions	628	1170	1643	2276	2721	3434
	Population Time (s)	4.079 (0.122)	4.503 (0.122)	4.981 (0.141)	5.572 (0.164)	5.883 (0.156)	6.794 (0.182)
Validation Phase	*Constraint Validation Time (s)*	6.357 (0.312)	10.532 (0.480)	14.300 (0.969)	16.064 (1.262)	26.008 (5.922)	37.596 (4.775)

Table 5.3: Perofrmance Evaluation Results I

The first experiment was carried out in three phases. In the first phase, we checked the consistency of the Tbox of the BPKB (*Consistency Phase*). In the second phase, we ran the population tool to transform an annotated BPD into an OWL Abox (*Population Phase*). In the last phase, we validate the BPKB against the constraint considered, to check whether the given process satisfies it or not (*Validation Phase*).

The reasoning tasks required in each phase have been performed with the support of the Pellet reasoner (v2.0.2)[14], integrated with the Pellet IC Validator (v0.4)[15] for the constraint validation tasks.

The results of this first experiment are reported in the lower half of Table 5.3[16]. As shown by the results, the constraint validation phase is the most expensive one: in this phase the computation time increases considerably as the size of the process (and, hence, of the BPKB) grows. As expected, the same trend (related to the size of the process to encode) is also exhibited by the population phase, though with a minor impact on the performance.

In the second experiment we considered a single process (`P4`, from the first experiment) and an increasing number (1, 5, 10, 50, and 100) of process constraints (of the same kind of requirement (e)). The purpose of this second experiment was to study the performance of the constraint validation phase as the number of process constraints grows. The results of

[13]We recall that checking the consistency of an $\mathcal{ALCHOIN}(\mathcal{D})$ ontology is an NExpTime-hard problem.

[14]`http://clarkparsia.com/pellet/`

[15]`http://clarkparsia.com/pellet/icv/`

[16] The time values reported in Tables 5.3 and 5.4 are in the form *avg* (*sd*), where *avg* and *sd* are respectively the arithmetic mean and the standard deviation of the execution times obtained over 100 runs on the same input data.

the second experiment are reported in Table 5.4[16]. As shown by the results, the number of

Number of Constraints	**1**	**5**	**10**	**50**	**100**
Validation Time (s)	16.147 (0.995)	16.245 (0.944)	16.017 (1.114)	16.197 (1.159)	16.177 (1.144)

Table 5.4: Performance Evaluation Results II

constraints to validate does not significantly impact on the performance of the constraint validation phase.

Overall the results show that the performance of the current (state of the art) tools is compatible with the modellers' needs and that these tools also allow an on-line usage on processes of small / medium size.

Chapter 6

Crosscutting Concern Documentation

"Most of the fundamental ideas of science are essentially simple, and may, as a rule, be expressed in a language comprehensible to everyone."
Albert Einstein

Beyond the process workflow itself, business processes usually involve several other concerns (i.e., "any matter of interest in a software system" [165]), often scattered across the whole process and tangled with the main view. A crosscutting concern in a business process is a process feature that cannot be modularised into a single unit (e.g., an activity or a sub-process), thus resulting scattered across the process and tangled with other concerns (either classic process units or other crosscutting concerns).

For example, several points in the workflow of a business process associated with online shopping may deal with user preferences (either collecting preferences or making suggestions based on them). However, the "user preferences" concern is not represented separately, documented explicitly or even searchable (except for pure textual search) in the business process. Consistent evolution of user preference management becomes hence troublesome and error prone. Whenever, for example, the user preference policy has to be changed or one of its occurrences scattered across the process is impacted by local changes in the process, documentation and knowledge related to the concern could support the designer in this difficult task.

Though allowing to represent classic process perspectives, existing business process modelling languages do not provide any constructs to describe crosscutting concerns. Knowledge about the crosscutting functionalities modelled in a process (e.g., preferences in an on-line purchase process or workflow patterns), in fact, mainly pertains to the semantic (domain-related) description rather than to the syntactic perspective of the business process. This lacking capacity in identification and explicit representation of crosscutting concerns demands for mechanisms devoted to their retrieval and documentation.

In this chapter we investigate how the knowledge provided by semantic annotations and the formality deriving from the process model encoding into the BPKB allow to retrieve and document, either by manually querying or semi-automatically mining, crosscutting concerns. In detail, in Section 6.1 we will introduce a visual language, BPMN VQL, for querying process models (and hence manually retrieving crosscutting concerns), while in Section 6.2 we will describe an approach to semi-automatically mine crosscutting concerns.

The material presented in this chapter has been published in [49].

6.1 Concern Querying

Business processes can be very large and retrieving information scattered across their flow is often resource and time-consuming. One option to retrieve business concerns (and hence also crosscutting concerns) is querying the process, i.e., matching the query asking for the desired concern against the process elements. The possibility of automatically querying processes and visualizing the retrieved results would be very useful for business designers and analysts in order to save their time and effort. The relevance of adequate means to query business processes has also been recognized by the Business Process Management Initiative (BPMI), that started the definition of a standard query language for business processes (BPQL) [84]. Querying business processes demands for languages able to specify process model characteristics and, at the same time, close enough to the knowledge of people working with process models. Many of the process query languages in the literature (e.g., [9, 16]), in fact, are visual languages, exploiting the process-like visual representation for expressing process model properties. In this trend of the languages for querying processes, we propose a visual language for BPMN processes, BPMN VQL (BPMN Visual Query Language). BPMN VQL syntax is close to the BPMN and it exploits the formal framework underlying semantically annotated processes for retrieving query results. In the next sections we first describe the BPMN VQL syntax and the mechanism allowing the query execution and then we provide an evaluation of the language in terms of expressive power with respect to the alternative visual languages and time performance.

6.1.1 BPMN VQL

The automatic querying of processes is an appealing possibility for business analysts and designers in many situations, e.g., when locating specific parts of the process, analysing a particular process concern, as well as retrieving parts of process presenting known characteristics. BPMN VQL is a query language able to quantify over BPMN business process elements, localize interesting concerns and, once identified, present them to the user by visually highlighting their occurrences in the BPMN BPD. Moreover, since a critical issue of the query language is usability (it is going to be used by business designers and analysts), the proposed language is a visual language, as close as possible to what business experts already know: BPMN itself.

Queries in BPMN VQL are built by using:

- standard BPMN graphical notation, to quantify over BPD objects;
- stereotypes for BPMN hierarchies of BPD graphical objects;
- semantic annotations and inference reasoning for BPD objects with a specific business domain semantics;
- composition of semantic annotations by means of the logical operators ($\wedge, \vee$ and $\neg$), to quantify over specific BPD objects or groups of objects with a precise business domain semantics;
- composition of more subqueries by means of the OR and/or the NOT operators;
- the transitive closure of direct connections and sub-process inclusions between BPD flow objects by means of the PATH operator, matching two BPD objects connected by at least one path in the BPD, and the NEST operator, matching activities nested in sub-processes;
- domain ontology relationships, by means of the DOR (Domain Ontology Relationship) operator (that allows to directly refer to a BDO relationship).

The language allows the description of queries with a structure similar to those formulated in SQL (Structured Query Language) for querying relational databases. In fact, it provides a different notation to distinguish between the "matching" part (*matching criterion*) of the query, that determines the criterion to match (i.e., the WHERE clause in an SQL query), and the "selection" part (*selection pattern*), that allows to visualize only the selected subpart of the matching result (i.e., the SELECT clause in an SQL query).

The components of the *selection pattern* have a darker background, thicker lines and bold font style.

Since BPMN language and syntactic matching allow to trivially obtain, via enumeration, every process subpart, including the whole process, the described visual language is complete. However, the ability to quantify and reason, allows to express queries in a more compact and concern-oriented form.

BPMN VQL queries are translated automatically into SPARQL (SPARQL Protocol and RDF Query Language) [72, 73] and executed by exploiting a SPARQL implementation. SPARQL, in fact, is an RDF-based query language, standardized by the World Wide Web Consortium [72] and widely accepted in the semantic web community (thus supported by several implementations). Its latest (draft) proposed version is the SPARQL 1.1. [73]: it enriches SPARQL 1.0. with new features[1]. In detail, we use the Jena[2] API for the query formulation and the ARQ[3] engine for the query execution.

The SPARQL translation of the BPMN VQL queries is based on the BPMNO and BDO ontologies and the query results are obtained by querying the BPKB populated with the BPD and its objects. The translation is obtained by: (i) requiring that each BPD graphical object in the visual query is an instance of the corresponding BPMNO class and each semantically annotated BPD object an instance of the BDO class corresponding to the annotation (e.g., in Figure 6.2, "$?t\ rdf:type$ BPMNO:task" and "$?t\ rdf:type$ BDO:to_check", respectively); (ii) constraining the BPD graphical objects in the query according to the corresponding BPMN structural properties (e.g., "$?as$:has_sequence_flow_source_ref $?t$" in Figure 6.2); (iii) using FILTER and EXISTS SPARQL constructs for realizing the NOT operator and the SPARQL UNION construct for the OR operator; (iv) using the SPARQL 1.1 property paths for composing ontology properties and/or denoting their transitive closure (e.g., "$?a1$ (BPMNO:has_sequence_flow_source_ref_inv /BPMNO:has_sequence_flow_target_ref)$*$ $?a2$" in Figure 6.7); (v) filling the SPARQL SELECT clause with variables representing the part of the query to be retrieved in the process (i.e., the darker or thicker graphical objects and the semantic annotations in bold).

In the following we describe in more detail the BPMN VQL, using examples that refer to the product assembly process shown in Figure 6.1 (the same used in Chapter 4). For each example, in order to formalize the query and give it a precise semantics, we also provide the translation of the query into SPARQL.

Queries using standard BPMN graphical notation. Single graphical objects

[1]A list of the working draft documents related to the new features of SPARQL is available at `http://www.w3.org/TR/#tr_SPARQL`

[2]`http://jena.sourceforge.net/`

[3]`http://jena.sourceforge.net/ARQ/`

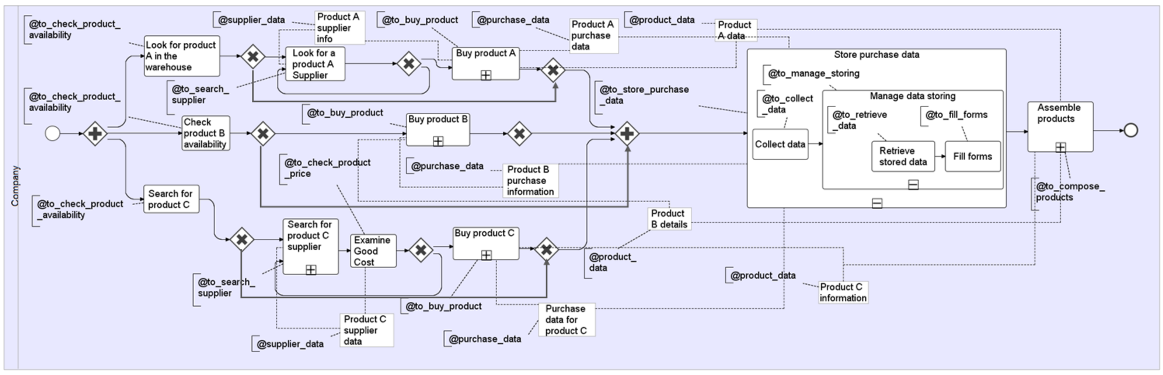
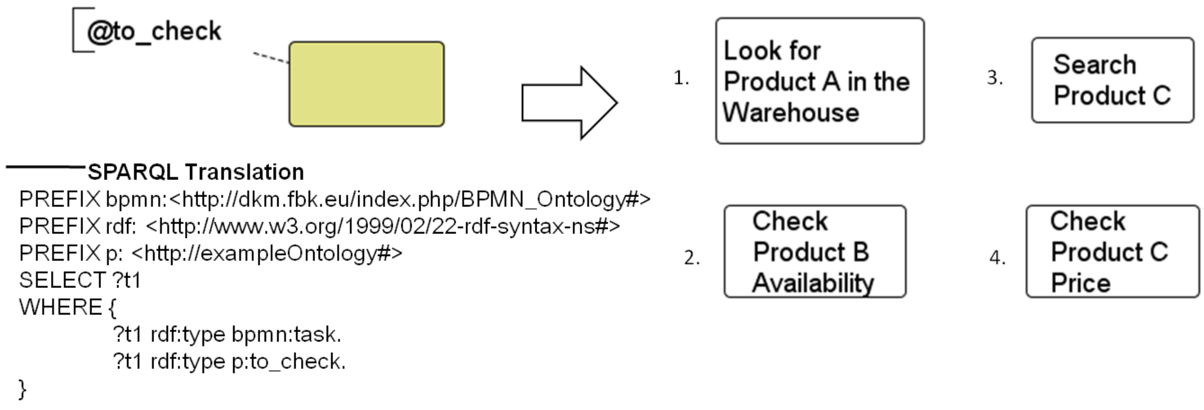


Figure 6.1: An example of a semantically annotated BPMN process.

in the BPMN notation (e.g., rounded rectangles, diamonds, arrows) are used to match either an instance with a specific label (if the BPD object in the query is labelled), or all the instances of the corresponding BPMNO class (if the BPD object in the query is unlabelled). In the first case, it is necessary to specify both the BPMNO-type of the BPD object and the label of the specific instance required. In the second case, it is sufficient to provide the specific BPMN object representation, without any label, thus indicating any instance of the specified BPMNO-type. However, the expressive power of the BPMN notation in the BPMN VQL is not limited to individual graphical objects. By composing together more BPD objects, in particular by linking flow objects and/or artefacts by means of connecting objects, it is possible to match whole subparts of the process.

Queries using stereotypes. Stereotypes are indicated within guillemets inside the BPMN activity symbols (i.e., rounded rectangles) and represent (sub-)hierarchies of BPD graphical objects.

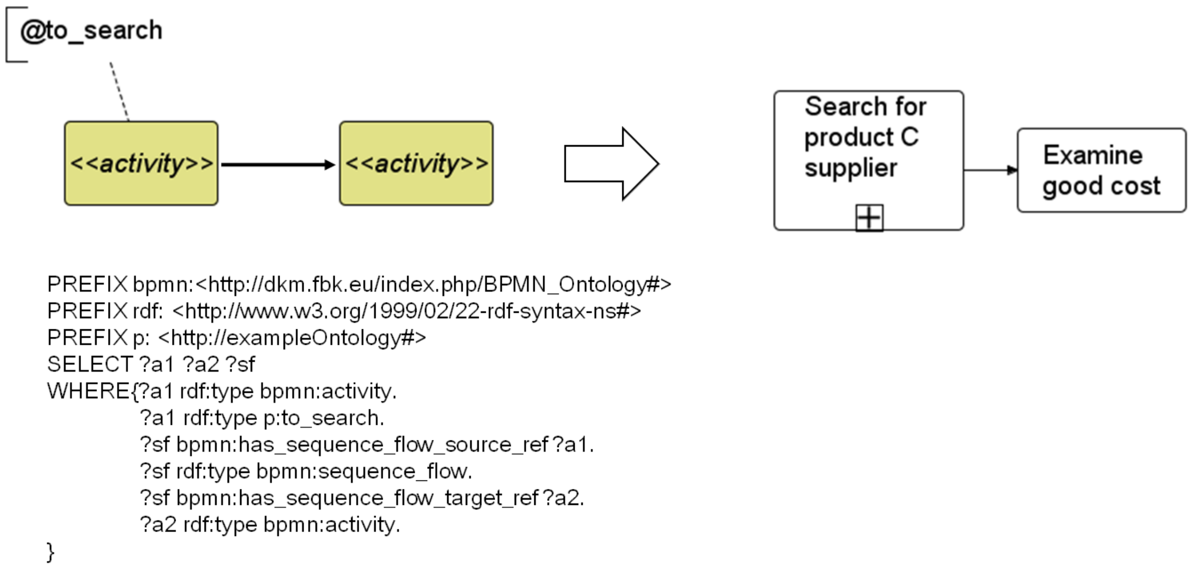
Queries using semantic annotation. Queries exploiting semantic annotations are used to select instances of the BPMNO, representing, directly (i.e., without inference) or indirectly (i.e., with inference), a specific ontological concept (i.e., of a given BDO-type). The BPMNO instances in the query result, hence, will also be instances of a class/superclass of the BDO ontology. In Figure 6.2, for example, we ask for all the tasks that check something. Although annotated with different (i.e., more specific) concepts, tasks are added to the result, as long as to_check is an ancestor of their annotations (e.g., their annotation being to_check_product_availability and to_check_product_price). Figure 6.3, instead, provides an example in which the standard BPMN VQL, stereotypes and semantic annotations are used together for retrieving all the pairs of directly connected activities, such that the source is a to_search activity, and for retrieving their connecting sequence flow. The result of the query, hence, contains only the triplet composed of the


@to_check
1.
Look for Product A in the Warehouse
3.
Search Product C
2.
Check Product B Availability
4.
Check Product C Price

SPARQL Translation

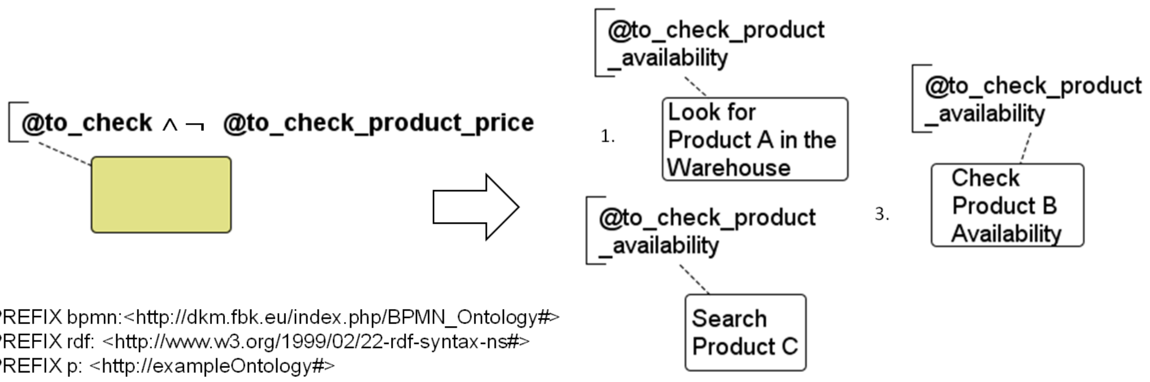
```
PREFIX bpmn:<http://dkm.fbk.eu/index.php/BPMN_Ontology#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX p: <http://exampleOntology#>
SELECT ?t1
WHERE {
        ?t1 rdf:type bpmn:task.
        ?t1 rdf:type p:to_check.
}
```

Figure 6.2: Example of a query using semantic annotations: it queries for all the tasks that check something.



```
PREFIX bpmn:<http://dkm.fbk.eu/index.php/BPMN_Ontology#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX p: <http://exampleOntology#>
SELECT ?a1 ?a2 ?sf
WHERE{?a1 rdf:type bpmn:activity.
        ?a1 rdf:type p:to_search.
        ?sf bpmn:has_sequence_flow_source_ref ?a1.
        ?sf rdf:type bpmn:sequence_flow.
        ?sf bpmn:has_sequence_flow_target_ref ?a2.
        ?a2 rdf:type bpmn:activity.
}
```

Figure 6.3: Example of a query using standard BPMN, stereotypes and semantic annotations: it queries for all the pairs of directly connected activities, such that the source is a searching activity, and for the sequence flow connecting them.


@to_check ∧ ¬ @to_check_product_price
@to_check_product _availability
1.
Look for Product A in the Warehouse
@to_check_product _availability
Search Product C
3.
@to_check_product _availability
Check Product B Availability

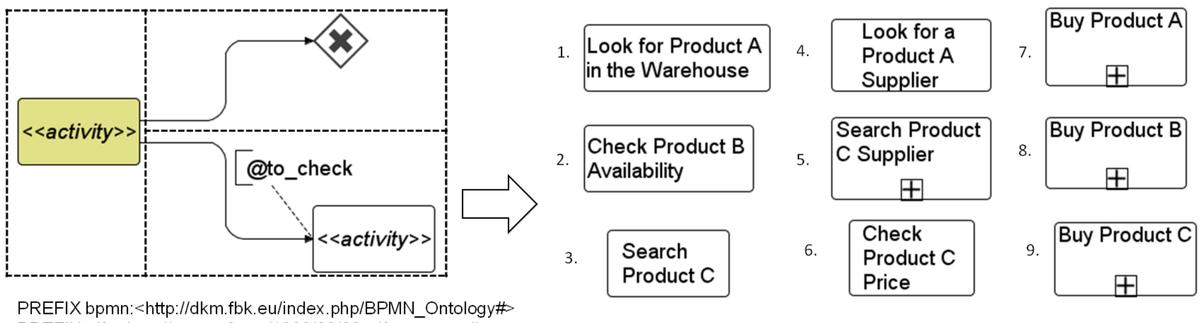
```
PREFIX bpmn:<http://dkm.fbk.eu/index.php/BPMN_Ontology#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX p: <http://exampleOntology#>
SELECT ?t1 ?as ?an
WHERE {?t1 rdf:type bpmn:task.
        ?t1 rdf:type p:to_check.
        FILTER (
               NOT EXISTS {?t1 rdf:type p:to_check_product_price}
        )
        ?as bpmn:has_connecting_object_target_ref ?t1.
        ?as rdf:type bpmn:association.
        ?as bpmn:has_connecting_object_source_ref ?an.
        ?an rdf:type bpmn:annotation.
}
```

Figure 6.4: Example of a query using logical operators for composing semantic annotations: it queries for all the tasks that check something, except the product price.

task labelled with “Search for product C supplier”, the task labelled with “Examine good cost” and for retrieving the sequence flow representing their direct connection.

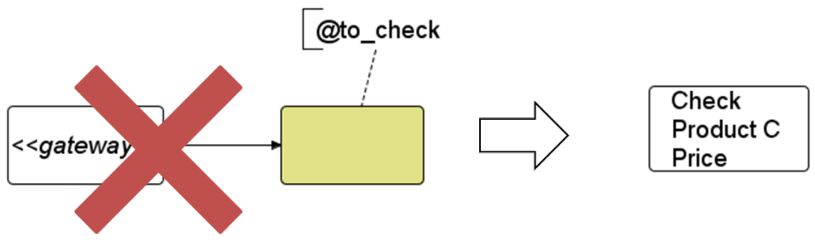
Queries using logical operators for semantic annotations. In order to compose queries involving annotations, the classic logical operators ($\wedge$, $\vee$, $\neg$) are used. Their semantics is the following one: the $\wedge$ (and) operator is a binary operator representing the intersection between two concepts, returning all the instances common to the two operands in the $\wedge$ expression. The $\vee$ (or) operator is another binary operator representing the union between two concepts, returning all the instances belonging to one or more operands in the $\vee$ expression. The $\neg$ (not) operator is a unary operator representing the negation of a concept, returning all the instances that do not belong to the negated set. In Figure 6.4, the $\wedge$ and the $\neg$ operators are composed together in order to select all the activities (both tasks and sub-processes) in the process that check something, except the product price. The result consists of the three tasks that check the product availability.

Queries using operators for composing subqueries. In order to be able to express more complex queries, BPMN VQL provides three operators for composing subqueries. The default operator between two or more subqueries is the intersection of the results provided by each subquery. Two more operators are introduced in order to support also


<<activity>>
@to_check
<<activity>>
1. Look for Product A in the Warehouse
2. Check Product B Availability
3. Search Product C
4. Look for a Product A Supplier
5. Search Product C Supplier
6. Check Product C Price
7. Buy Product A
8. Buy Product B
9. Buy Product C

```
PREFIX bpmn:<http://dkm.fbk.eu/index.php/BPMN_Ontology#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX p: <http://exampleOntology#>
SELECT ?a1
WHERE {?a1 rdf:type bpmn:activity.
        ?sf1 bpmn:has_sequence_flow_source_ref ?a1.
        ?sf1 rdf:type bpmn:sequence_flow.
        ?sf1 bpmn:has_sequence_flow_target_ref ?a2
        {?a2 rdf:type bpmn:activity.
         ?a2 rdf:type p:to_check.}
         UNION
        {?a2 rdf:type bpmn:data_base_exclusive_gateway.}
}
```

Figure 6.5: Example of a query using the OR operator for composing two or more subqueries: it queries for all the activities connected to a gateway or to another activity checking something.



```
PREFIX bpmn:<http://dkm.fbk.eu/index.php/BPMN_Ontology#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
 PREFIX p: <http://exampleOntology#>
SELECT ?t1
WHERE {?t1 rdf:type bpmn:task.
        ?t1 rdf:type to_check
         FILTER (
                 NOT EXISTS {?sf1 bpmn:has_sequence_flow_target_ref ?t1.
                             ?sf1 rdf:type bpmn:sequence_flow.
                             ?sf1 bpmn:has_sequence_flow_source_ref ?g1.
                             ?g1 rdf:type bpmn:gateway.}
          )
}
```

Figure 6.6: Example of a query using the NOT operator for composing two or more subqueries: it queries for all the tasks that check something and that are not preceded (via sequence flow) by any gateway.

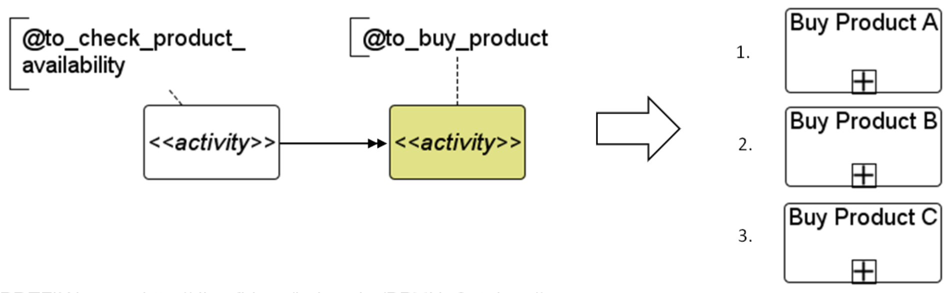
the union and the negation of subquery results. The OR operator is depicted as a dotted table listing all possible alternative subparts of the query to match. The query in the example in Figure 6.5 asks for all the instances of any activity followed by an exclusive gateway or by an activity that checks something. The result is provided by the eight activities preceding the six exclusive gateways and by the "Search Product C Supplier" sub-process preceding the "Check Product C Price" task. The NOT operator is depicted as a cross over the negated (set of) BPD object(s). The query in Figure 6.6, for example, looks for all the tasks that check something but that are not preceded by any kind of gateway. Therefore in the result, the three tasks that are instances of the BDO class to_check_product_availability are discarded, while only the "Check Product C Price" task is reported.

Queries using the transitivity operators. In order to ensure users a higher navigability of process models, two operators supporting transitivity have been introduced: the PATH and the NEST operators.

The PATH operator allows to match paths connecting two BPD flow objects (of the same level of nesting). It is depicted as a BPMN sequence flow but with two heads, thus symbolizing any intermediate graphical object (both flow objects and sequence flows) encountered along the path. The query in Figure 6.7, for example, asks for all the activities that buy a product and for which there exists a sequence flow path starting from a to_check_product_availability activity and reaching them. The result of this query applied to the product assembly process consists of the three sub-processes annotated by to_buy_product, since there exists at least a path from the "Check Product A In The Warehouse" task to each of them.

The NEST operator, instead, allows to capture BPD graphical objects nested at any level of depth in sub-processes. It is depicted as a small oblique arrow in the upper right corner of the sub-process and with the head pointing to the external part of the sub-process. The query in Figure 6.8, for example, retrieves all the to_retrieve tasks directly or indirectly contained in sub-processes storing purchase data. The result is provided by the task "Retrieve stored data" contained in the sub-process "Manage data storing", in turn contained in the to_store_purchase_data sub-process.

Queries using the DOR Operator. Sometimes it might be useful to be able to express also domain ontology relationships for querying specific business domain concerns. In order to allow users to formulate a query involving a domain ontology relationship, an operator has been introduced in the BPMN VQL: the DOR operator. It is depicted as a dashed arrow connecting graphical objects and/or semantic concepts and it represents a



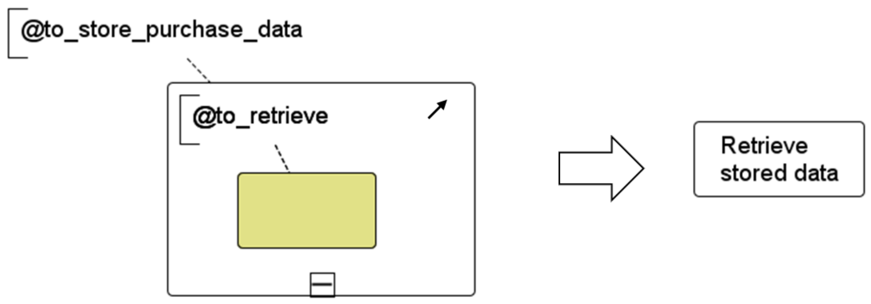
```
PREFIX bpmn:<http://dkm.fbk.eu/index.php/BPMN_Ontology#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX p: <http://exampleOntology#>
SELECT ?a1 ?a2
WHERE { ?a1 rdf:type bpmn:activity.
        ?a1 rdf:type p:to_check_product_availability.
        ?a1 (bpmn:has_sequence_flow_source_ref_inv/bpmn:has_sequence_flow_target_ref)* ?a2.
        ?a2 rdf:type bpmn:activity.
        ?a2 rdf:type p:to_buy_product.
}
```

Figure 6.7: Example of a query using the PATH operator: it queries for all the activities that buy products and for which there exists at least a path, consisting of sequence flows, that connects a to_check_product_availability activity to the current activity.

domain ontology relationship. For example, the query in Figure 6.9, looks for all the pairs of instances of data objects, whose first component refers to supplier and whose second component concerns any of the supplied products. The provides relationship is a domain relation between the instances of the two semantic concepts, supplier and product, respectively. The other two DOR operators labelled has_specifier represent the domain ontology relationships between the pairs of data objects' BDO classes (in this case derived from the BDO classes supplier_data and product_data) and their specifiers (supplier and product, respectively). In the example shown in Figure 6.9, the two pairs of activities ("Product A Supplier Info", "Product A Data") and ("Product C Supplier Data", "Product C Information") are reported in the result.

6.1.2 BPMN VQL Evaluation

In this subsection we provide a first evaluation of the BPMN VQL in terms of functionality provided with respect to similar process query languages and in terms of time performance. A further evaluation of BPMN VQL (related to its ease of use), carried out by means of an empirical study, is described in Chapter 8.



```
PREFIX bpmn:<http://dkm.fbk.eu/index.php/BPMN_Ontology#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX p: <http://exampleOntology#>
SELECT ?a1 ?sb1
WHERE {?a1 rdf:type bpmn:activity.
        ?a1 rdf:type p:to_retrieve.
        ?sb1 bpmn:has_embedded_sub_process_sub_graphical_elements+ ?a1.
        ?sb1 rdf:type bpmn:embedded_sub_process.
        ?sb1 rdf:type p:to_store_purchase_data.
}
```

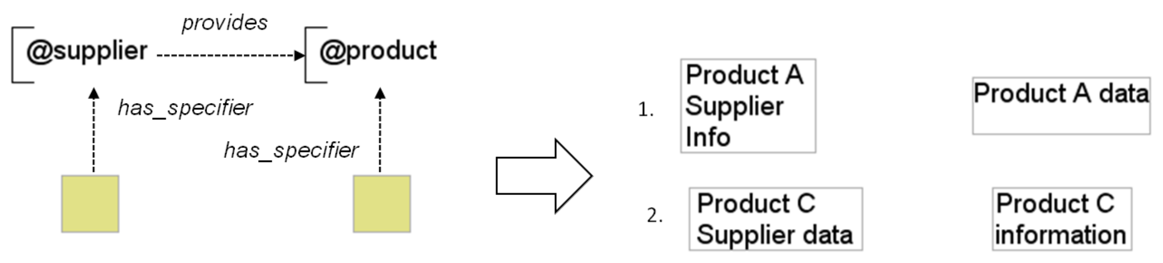
Figure 6.8: Example of a query using the NEST operator: it queries for to_retrieve tasks directly or indirectly contained in sub-processes storing purchase data.

BPMN VQL Functionality Evaluation

Several process query languages, including BPMN VQL, have been proposed in the literature (e.g., [120, 119, 16, 9, 49]) for querying processes and process repositories. Though all of them advocate the need to be "easy" to use, they implement their objective in different ways. In detail, they can be classified in two main categories: the textual and the visual one.

BPQL [120], a language based on the Stack Based Query Language (SBQL) [164], mainly used to retrieve (and manage) information with specific characteristics related not only to the process structure, but also to execution objects and performers, belongs to the first class. Similarly, the language proposed by Missikoff et al. [119] falls in the first group. Beyond the process design, it also deals with the execution level (by allowing to query process traces) and the orthogonal dimension of the business ontology (by querying about business aspects). It presents to users a syntax similar to the SQL (i.e., a textual syntax), that is then translated into Prolog rules [104].

In the second category, to the best of our knowledge, beyond BPMN VQL, two other languages exist: BP-QL [16] and BPMN-Q [9]. All these three visual languages for querying processes are based on graph matching. However, each of them has peculiarities that makes it different from others. BP-QL is a language for querying BPEL processes,



```
PREFIX bpmn:<http://dkm.fbk.eu/index.php/BPMN_Ontology#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX p: <http://exampleOntology#>
SELECT ?do1 ?do2
WHERE {?do1 rdf:type bpmn:data_object.
          ?do1 rdf:type  ?s.
          ?s p:has_specifier p:supplier.
          ?do2 rdf:type bpmn:data_object.
          ?do2 rdf:type ?p.
          ?p p:has_specifier p:producer.
           ?s p:provides ?p.
}
```

Figure 6.9: Example of a query using the DOR operator: it queries for all the pairs of data object instances representing data of pairs of suppliers and products connected by a BDO provides relationship.

while the others use BPMN. Moreover, the BP-QL and the BPMN-Q are intended to be languages for mining process repositories, while BPMN VQL is mainly intended to be used for querying a single process [11]. BPMN VQL and BPMN-Q are both thought to be specification languages, but BPMN-Q can also be used as an execution language.

In detail, BP-QL is based on business process patterns, that allow to describe the desired control-flow or data flow pattern of interest. It enables the navigation along two axes (the *path-based* and the *zoom-in* axis), thus allowing users to have paths in query results, as well as to control the granularity in business processes. The BP-QL implementation exploits the graph matching functionality of XML and is based on Active XML (AXML)[4], an XML enriched with service calls to Web services.

Similarly, BPMN-Q allows to express structural BPMN queries and to query repositories of business process models. It also hides the query complexity behind a visual interface and exploits the graph matching for the query execution by following a step-by-step procedure that incrementally binds the query graph to one or more process graphs (i.e., the query results). However, BPMN-Q also provides special constructs and mechanisms to abstract over graph nodes both structurally and semantically, by applying information

[4]http://activexml.net

Feature		**BP-QL**	**BPMN-Q**	**BPMN VQL**
Type of Flow	*Design-time Control Flow*	+	+	+
	Design-time Data Flow	+	+	+
	Business Flow	-	-	+
Abstraction	*Node Type abstraction*	-	+	+
	Edge Abstraction	+	+	+
	Node Label Abstraction	-	+	+
	Node Negation	+	-	+
	Edge Negation	+	+	+
	Abstraction over hierarchical structures (i.e., sub-processes)	+	-	+
Projection	*Projection*	+	-	+
Usage scenarios	*Process search in repositories*	+	+	-
	Single Process Querying	+	+	+
	Compliance Checking and Anomaly Detection	-	+	-

Table 6.1: Feature-based comparison among three visual query languages for business processes

retrieval techniques, as the notion of similarity in the enhanced topic-based vector space model, eTVSM [96].

We extended the work by Awad et al. [11] that analyses differences and similarities between BP-QL and BPMN-Q based on language features. In detail, we refined the characteristics of the languages identified by the authors and, in the comparison, we analysed the BPMN VQL too. Table 6.1 summarizes the performed analysis.

Similarly to Awad et al. [11], we identified four main characteristics of process query languages: (i) the type of flows they allow to query; (ii) their capability of abstraction; (iii) their capability of projecting the retrieved results; and (iv) the envisaged usage scenarios.

With respect to the first feature, all the three query languages allow to query the control and the data flows of business process models, though BPMN-Q is able to query not only direct but also indirect associations of data and activities. However, only BPMN VQL allows to query the business flow. This capability is particularly helpful for business analysts, for example when they need to retrieve the business relationship(s) existing among business objects, as well as to locate the business objects involved in a business relationship.

Among the characteristics of the abstraction category, all the three languages allow to abstract over edges (i.e., to represent paths, as well as to negate edges). Only BPMN-Q and BPMN VQL allow to abstract over node types and node labels, though BPMN-Q, differently from BPMN VQL, always abstracts over node labels, i.e., it does not allow the exact matching of a label. Finally, only BP-QL and BPMN VQL allow to negate a given node. The language versatility, i.e., its capability to allow, besides the exact matching, the abstraction over process nodes, labels and edges (i.e., to actually support the features in the "abstraction" category), makes the language very useful for designers, using it in different circumstances, i.e., both when they have a specific knowledge of the target of their queries and also when they are not precisely aware about the searched concern.

In the third category, the capability of the languages to project specific parts of the matched pattern is considered. Both BP-QL and BPMN VQL provide mechanisms for specifying the selected part of the pattern, while BPMN-Q always returns the whole pattern. This capability is of great importance for business people, for example when a precise and quick-to-visualize answer is required for analysis purposes or when the query results have to be provided as input parameters to other services.

Finally, in the fourth group, the envisaged usage scenarios are investigated. While BPMN VQL aims at querying a single process model, among the goals of BP-QL and BPMN-Q there is also the capability of querying repositories of business processes in order to retrieve interesting processes or parts of processes. Finally, Awad et al. [11] also envisage a third use for BPMN-Q: the compliance checking and the anomaly detection.

Summarizing, BPMN VQL presents the most complete set of abstraction and projection mechanisms, while the list of usage scenarios envisaged for the language does not include querying repositories and verifying constraints. However, though BPMN VQL has been conceived as a language for retrieving interesting concerns in single process models, its use to query repositories of process models and to verify constraints is quite straightforward. We included these features in the table to be coherent with the comparison provided by Awad et al. [11], though they are out of the scope of our analysis.

BPMN VQL Performance Evaluation

We performed an experiment in order to provide a first evaluation of the performance of the BPMN VQL. The time of query answering, in fact, is a critical factor for business analysts and designers. We hence performed a preliminary experiment to evaluate whether BPMN VQL queries have a reasonable response time.

In the experiment[5] we considered six different processes (the same used for constraint checking evaluation in Chapter 5) of increasing size (with a number of process graphical elements ranging from 92 to 475), and, for each of them, a set of seven queries each aimed at investigating a different construct of the language. BPMN VQL queries were translated into SPARQL 1.1 queries and executed by means of the SPARQL ARQ implementation[6]. The purpose of the experiment was to study the performance of BPMN VQL as the size of the BPKB (in terms of instances) grows and as the structure (and hence the complexity) of

[5]The machine used for the experiment is a desktop PC with an Intel Core i7 2.80GHz processor, 6 Gb of RAM, and running Linux RedHat.

[6]`http://jena.sourceforge.net/ARQ/`

queries changes. The number of BPD graphical objects as well as the main characteristics and the DL expressivity of the domain ontologies used to annotate the processes are listed in the top rows of Table 6.2, while a reference to an example query similar to the one used in this experiment is shown among brackets next to each query (in the first column of Table 6.2). Each of the seven types of queries considered in the experiment, in fact, has the same structure of one of the examples introduced in Subsection 6.1.1 for presenting the different BPMN VQL operators. In detail, the first query ($Q1$) looks for tasks of a given business domain type (i.e., similarly to the query in Figure 6.2); the second ($Q2$) retrieves direct connections between pairs of flow objects, where the domain type of the first one is specified (similarly to the query in Figure 6.3); the third ($Q3$) investigates the use of logical operators for the composition of semantic annotations (similarly to the query in Figure 6.4); the fourth ($Q4$) makes use of the OR operator (similarly to the query in Figure 6.5); the fifth ($Q5$) contains the NOT operator (similarly to the query in Figure 6.6); the sixth ($Q6$) analyses the PATH operator (similarly to the query in Figure 6.7); and, finally, the seventh ($Q7$) investigates the use of the NEST operator (similarly to the query in Figure 6.8).

The results of the experiment are reported at the bottom of Table 6.2[7]. Times related to query executions have been collected after an ontology preprocessing phase, in which the inferred model has been computed by the Pellet[8] reasoner[9].

As expected, not only the time required for loading the ontology increases when the process size grows (sixth row in the Table 6.2), but also the time used for the ontology preprocessing and for query execution (ranging from an average of 0.007 seconds for the process with the smallest size to about 0.013 seconds for the process containing 475 process elements). On the contrary, the type of the query does not significantly impact the performance of query execution, though minor differences among the considered types of queries exist. The largest amount of time was taken by the query using logical operators (both and and not operators in $Q3$) for the composition of semantic annotations ($Q3$), and the one exploiting the OR operator for the composition of subqueries ($Q4$). The cheapest queries in terms of time, instead, are $Q5$ and $Q7$, i.e., the query using the NOT and the NEST operator, respectively. All types of queries, however, complete their run in a very limited time, though a quite significant time is spent for ontology preprocessing (around 35 seconds in case of the largest process). The ontology preprocessing, carried on in this case study, is useful when the ontology is rarely modified. When, instead,

[7]The time values are expressed in seconds in the form *avg* (*sd*), where *avg* and *sd* are respectively the arithmetic mean and the standard deviation of the execution times obtained over 100 runs on the same input data.

[8]`http://clarkparsia.com/pellet/`

[9]The preprocessing phase includes also the time for structure construction required by the execution of the first query.

	P1	**P2**	**P3**	**P4**	**P5**	**P6**
Process Graphical Objects	92	175	237	327	387	475
DL Expressivity	$\mathcal{ALC}$	$\mathcal{ALC}$	$\mathcal{AL}$	$\mathcal{ALC}$	$\mathcal{AL}$	$\mathcal{ALC}$
Classes	124	124	101	114	79	124
Class Axioms	133	133	101	113	77	133
Ontology Loading Time (s)	1.445(0.033)	1.476(0.037)	1.498(0.037)	1.513(0.038)	1.528(0.037)	1.564(0.040)
Ontology Preprocessing Time (s)	4.459(0.999)	8.318(0.373)	13.090(0.892)	15.298(2.935)	37.237(14.238)	35.349(6.393)
Q1 (Figure 6.2)	0.003(0.000)	0.004(0.000)	0.012(0.002)	0.006(0.000)	0.012(0.001)	0.008(0.001)
Q2 (Figure 6.3)	0.004(0.000)	0.004(0.000)	0.006(0.000)	0.005(0.000)	0.006(0.001)	0.006(0.001)
Q3 (Figure 6.4)	0.017(0.001)	0.019(0.001)	0.020(0.001)	0.021(0.001)	0.021(0.002)	0.023(0.002)
Q4 (Figure 6.5)	0.011(0.001)	0.014(0.001)	0.018(0.001)	0.022(0.002)	0.025(0.003)	0.030(0.003)
Q5 (Figure 6.6)	0.003(0.001)	0.004(0.000)	0.004(0.000)	0.005(0.000)	0.005(0.002)	0.006(0.002)
Q6 (Figure 6.7)	0.009(0.000)	0.010(0.001)	0.009(0.001)	0.011(0.001)	0.012(0.002)	0.012(0.001)
Q7 (Figure 6.8)	0.003(0.001)	0.003(0.000)	0.005(0.000)	0.004(0.000)	0.005(0.000)	0.005(0.000)
Query Average Time (s)	0.0071	0.0083	0.0106	0.0106	0.0123	0.0129

Table 6.2: BPMN VQL performance

	P1	**P2**	**P3**	**P4**	**P5**	**P6**
Query Average Time (s)	0.1256	0.4769	0.9806	1.5899	3.0777	4.4373
Min Query Time (s)	0.034	0.083	0.138	0.206	0.295	0.411
Max Query Time (s)	0.292	0.891	1.817	2.922	5.763	8.492

Table 6.3: BPMN VQL performance on non-preprocessed ontologies

frequent changes occurs in the ontology, queries can be directly executed on the original ontology, on which no reasoning is applied before query execution. We collected the query execution performance also in this case and reported the average values in Table 6.3. As expected, query execution on non-preprocessed ontology takes more time than on the inferred ontology and it increases as the process size grows. However, the average and the worse response times (4 and 8 seconds, respectively, for the largest process) are still reasonable to be used in activities involving human interaction.

Given the results related to the BPMN VQL performance in this initial evaluation, we can state that the use of the language for querying processes is compatible with business designers' and analysts' needs and hence confirm its applicability as a means for supporting their work in retrieving business concerns in process models.

6.2 Crosscutting Concern Mining

Automatically querying business processes is an interesting avenue for business analysts and designers in order to retrieve interesting business concerns. However, the manual identification of crosscutting concerns, especially in large processes, could be non-exhaustive: acquiring knowledge about the concerns of interest for querying the process may be hard and expensive. For example, when a change occurs locally in the process, other instances of the same crosscutting concern could be directly or indirectly impacted too, thus requiring a consistent change across all the occurrences. It is likely, however, that the analyst is unaware of the crosscutting nature of concern occurrences involved in the change, hence a time-consuming analysis could be required for propagating the change to other concern occurrences, scattered across the process.

In order to support analysts in the identification of crosscutting concerns in business processes, we propose an approach for mining concerns in a semi-automatic way. Business domain knowledge enriching business processes by way of semantic annotations can, in fact, be exploited in order to mine candidate crosscutting concerns by analysing the occurrence of concepts used as annotations through Formal Concept Analysis, a technique for data analysis. The list of the retrieved candidate crosscutting concerns, ranked by level of scattering in the business process, is presented to the user in order to be further assessed and manually investigated.

We envisage at least three uses in which this semi-automatic mining of crosscutting concerns and their explicit documentation can be particularly helpful: (1) in process comprehension: it provides additional views on specific process concerns, thus supporting analysts in the comprehension of existing business processes going beyond the control flow view; (2) in the evolution phase: it allows to collect and document critical concerns requiring a separate and specific analysis in case of changes, thus supporting business analysts' tasks such as the location of changes scattered across the process and impact analysis; (3) during the transition to the implementation phase: by providing a view on crosscutting concerns, it supports the developers' work on different concerns (e.g., which concern has to be developed before/after another and what components or other concerns it affects).

6.2.1 Crosscutting Concern Mining

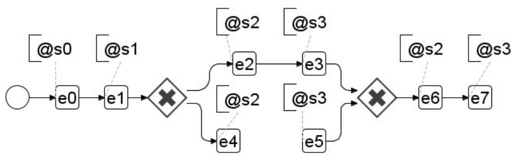
FCA is a branch of lattice theory used to build a lattice of *FCA-concepts*[10] (i.e., maximal groups of objects sharing common attributes) starting from a given context.

We apply FCA in order to find business domain concerns which crosscut multiple business process elements. This is achieved by searching for maximal groups of instances of process elements sharing common semantic concepts. Such maximal groupings are obtained as the FCA-concepts computed for a context $C = (E, S, R)$, where E is the set of instances of process elements, S the set of semantic concepts of the business domain and $R \subseteq E \times S$, the relation specifying that a given business process element instantiates a semantic concept of the business domain, as well as all its superconcepts in the ontology hierarchy of the business domain. If $(e, s) \in R$, e is said to be annotated by s. An FCA-concept c is a pair of sets (X, Y) where X, the *extent* of the FCA-concept, is defined as $X = \{e \in E | \forall s \in Y : (e, s) \in R\}$, while Y, the *intent* of the FCA-concept, is defined as $Y = \{s \in S | \forall e \in X : (e, s) \in R\}$. Intuitively, an FCA-concept is any maximal set of process elements associated with a maximal set of semantic concepts (including superconcepts) they instantiate.

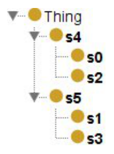
An FCA-concept $c_0 = (X_0, Y_0)$ is an FCA-subconcept of the FCA-concept $c_1 = (X_1, Y1)$ $(c_0 \sqsubseteq c_1)$ if $X_0 \subseteq X_1$ (or, equivalently, $Y_1 \subseteq Y_0$). The containment relationship between the FCA-concept extents (or intents), determines a partial order relationship. It is possible to show that this relationship defines a lattice [64].

Figure 6.10 shows the input context in tabular form (Figure 6.10(c)) for the BPMN process in Figure 6.10(a), whose elements are annotated by semantic concepts taken from the ontology in Figure 6.10(b). A relationship between a process element and a semantic concept exists if the element is annotated by the semantic concept itself or by one of its subconcepts. The set of FCA-concepts (Figure 6.10(d)) for such a context can be obtained by applying available tools (e.g., ToscanaJ[11]) implementing a concept analysis algorithm [64]. In Figure 6.10(e) we show the associated concept lattice, representing the sub-concept relationship as parent-child edges. Labels in the lattice depict the most generic (specific) node with a semantic annotation (process element) in the intent (extent), meaning that all downward (upward) reachable nodes have the same annotation (element) in their intent (extent). This is known as the *sparse labelling* of the concept lattice. For example, concept c_0 is labelled only by s_0 and e_0 in the lattice. The other element of its intent, s_4, is "inherited" from its parent in the lattice.

[10]In order to distinguish FCA concepts from ontology concepts we will always use FCA-concept in the former case.
[11]`http://toscanaj.sourceforge.net`



(a)



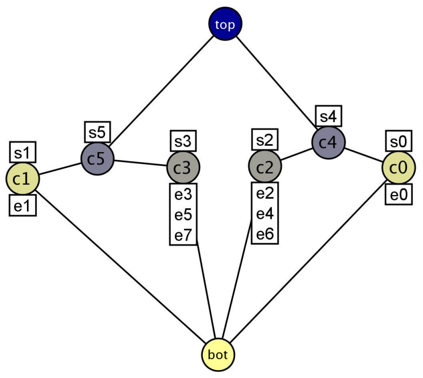
(b)

	s_0	s_1	s_2	s_3	s_4	s_5
e_0	√				√	
e_1		√				√
e_2			√		√	
e_3				√		√
e_4			√		√	
e_5				√		√
e_6			√		√	
e_7				√		√

(c)

top	$(\{e_0, e_1, e_2, e_3, e_4, e_5, e_6, e_7\}, \emptyset)$
c_5	$(\{e_1, e_3, e_5, e_7\}, \{s_5\})$
c_4	$(\{e_0, e_2, e_4, e_6\}, \{s_4\})$
c_3	$(\{e_3, e_5, e_7\}, \{s_3, s_5\})$
c_2	$(\{e_2, e_4, e_6\}, \{s_2, s_4\})$
c_1	$(\{e_1\}, \{s_1, s_5\})$
c_0	$(\{e_0\}, \{s_0, s_4\})$
bot	$(\emptyset, \{s_0, s_1, s_2, s_3, s_4, s_5\})$

(d)



(e)

Figure 6.10: FCA-concepts (Figure 6.10(d)) and concept lattice (Figure 6.10(e)) for the context table in Figure 6.10(c) extracted from the process in Figure 6.10(a) annotated by concepts from the ontology in Figure 6.10(b).

Looking at their position in the lattice, FCA-concepts with large extent (business process BPMN elements) share few semantic concepts (in the intent), hence appearing high in the lattice (generic FCA-concepts). FCA-concepts with large intent (semantic concepts) have few instances of process elements sharing all those semantic concepts (small extent), hence appearing down in the lattice (specific FCA-concepts). The bottom FCA-concept contains those elements that instantiate all the semantic concepts in the ontology and the top FCA-concept those semantic concepts that describe all the BPMN process instances.

By analysing the FCA-concept lattice, it is possible to find candidate business domain concerns that crosscut the BPMN process. The extent of each FCA-concept in the lattice gives an immediate intuition about the level of scattering involved. Crosscutting concern mining is achieved by ranking FCA-concepts according to the size of their extent (scattering). High scattering indicates the potential existence of a crosscutting concern.

Inspection of the semantic concepts in the FCA-concept intent gives clues for a semantic interpretation of the concern candidate. The shared semantic concepts of a candidate crosscutting concern are used by the business analyst for evaluating whether the candidate crosscutting concern actually represents a relevant business crosscutting concern. Visually, all BPD graphical objects in the extent of an FCA-concept are highlighted (through colours) whenever the given FCA-concept is selected by the user (through mouse click) in the lattice (see Figure 6.12).

In the example in Figure 6.10, c_4 and c_5 are the first two FCA-concepts found by ranking the FCA-concepts according to their extent size. Their scattering level is 4, since each of them has four process elements in the extent. Their meaning is provided by their semantic annotations, s_4 and s_5, respectively. Immediately down in the ranked list we encounter c_2 and c_3 with scattering equal to 3. Their meaning is the conjunction of the two semantic concepts that label them: $s_2 \wedge s_4$ and $s_3 \wedge s_5$ respectively.

Summarizing, the steps to follow in order to mine crosscutting concerns in business processes are the following:

1. FCA context construction based on the relationships between process elements and the semantic concepts they instantiate;
2. FCA lattice construction;
3. FCA-concept ranking according to decreasing scattering levels (i.e., extent sizes);
4. FCA-concept filtering according to a scattering threshold (i.e., the minimum extent size for an FCA-concept representing a candidate crosscutting concern);
5. semantic evaluation of the FCA-concept intent by the business analyst in order to discriminate whether it represents a relevant business concern or not.

6.2.2 Crosscutting Concern Mining Evaluation

The proposed technique has been applied to two case studies: an on-line purchase process and a process for resolving issues through e-mail votes. We evaluated the accuracy of the results obtained by applying the technique (i.e., steps 1 to 4 described in Subsection 6.2.1) to the two case studies, by resorting to two metrics of accuracy widely used in Information Retrieval [62]. In detail, for each of the two case studies, we computed precision and recall, as well as their combination, the F-Measure. *Precision* measures the proportion of candidate concerns reported by the proposed crosscutting concern mining technique that are judged as good (interesting and meaningful) candidates by an expert. *Recall* measures the proportion of concerns reported by the technique among all concerns that are judged of interest. *F-Measure* is the geometric average between precision and recall:

- *Precision = Reported and correct / Reported*
- *Recall = Reported and correct / To be reported*
- *F-Measure*= $\frac{2*precision*recall}{precision+recall}$

To determine precision, recall, and F-Measure for our case studies, we have carefully analysed the FCA-concept lattices, built on the basis of the relationships between their process elements and the respective annotations, to find out which reported FCA concepts (above scattering threshold) are correct and which ones are correct but unreported. Such assessment involves inevitably some degree of subjectivity for the identification of the "correct" crosscutting concerns (i.e., the gold standard). In order to deal with this subjective judgement, we adopted a guideline. In detail, to determine the gold standard crosscutting concerns, for each FCA-concept in the FCA-concept lattice we answered the following question: *In the context of the given process, is the FCA-concept a business concern for the analyst?*

On-line Purchase Case Study

The first case study (shown in Figure 6.11) is a generic on-line shopping process, obtained by looking at various existing on-line shopping Web sites and abstracting the underlying process into the common workflow. Two pools represent the customer and the on-line shop respectively. They repeatedly communicate by means of events generated by the customer's choices (e.g., product browsing, product search and cart management), until the customer asks for the checkout. This request leads to the control flow described inside the checkout sub-process. Activities are annotated with semantic information, i.e., concepts taken from a domain ontology, added to the process by means of the standard BPMN textual annotation. The process contains 23 top-level (i.e., not contained in sub-processes) activities, divided into 20 top-level tasks and 3 sub-processes, 14 top-level gateways and 13 top-level events. The domain ontology used for annotating the BPMN process contains 81 concepts and 29 of them have been explicitly used for annotating the process elements.

In applying the proposed mining technique to this process we used abbreviations for both the process elements (e.g., AC = Activity on Customer's pool; AS = Activity

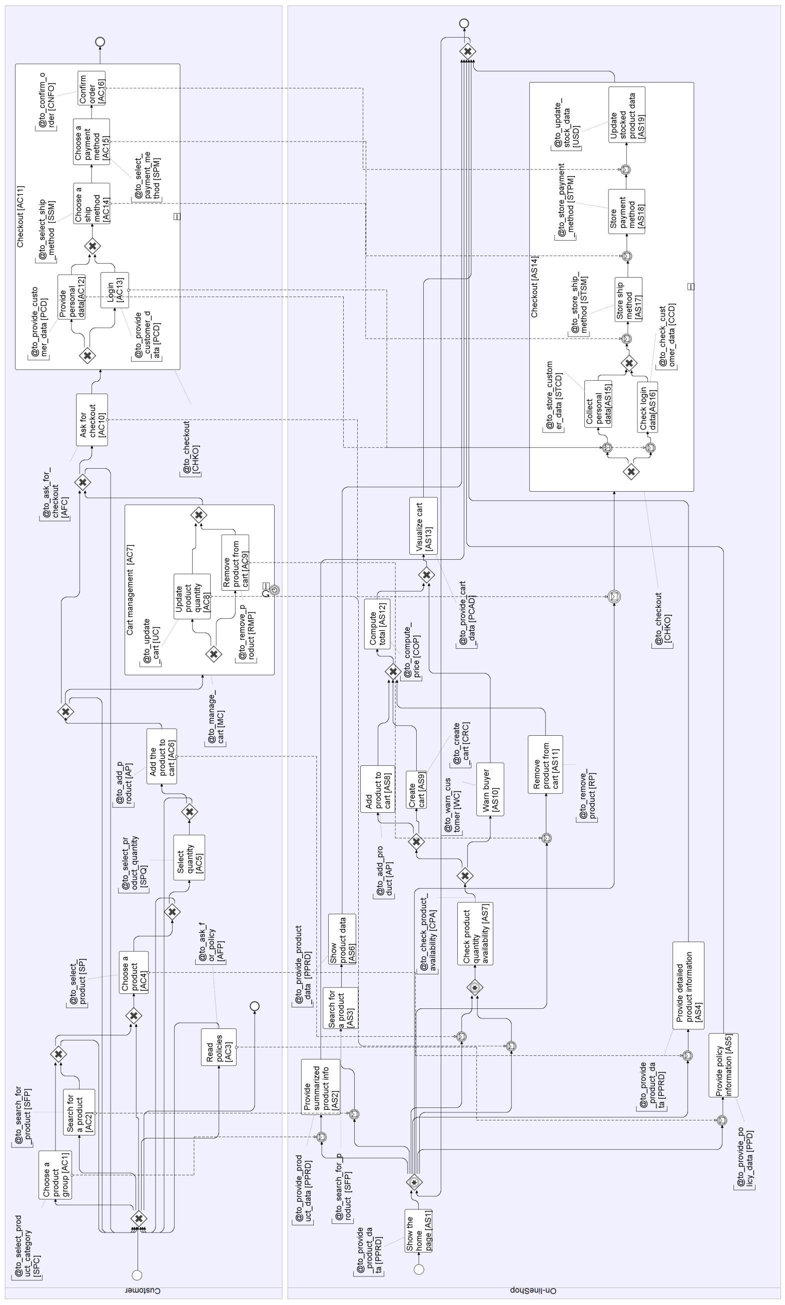

Customer
Checkout [AC11]
@to_confirm_order [CNFO]
Confirm order [AC16]
Choose a payment method [AC15]
@to_select_payment_method [SPM]
Choose a ship method [AC14]
@to_select_ship_method [SSM]
@to_provide_customer_data [PCD]
Provide personal data[AC12]
Login [AC13]
@to_provide_customer_data [PCD]
Ask for checkout [AC10]
@to_checkout [CHKO]
@to_ask_for_checkout [AFC]
Cart management [AC7]
Update product quantity [AC8]
Remove product from cart [AC9]
@to_update_cart [UC]
@to_remove_product [RMP]
@to_manage_cart [MC]
Add the product to cart [AC6]
@to_add_product [AP]
Select quantity [AC5]
@to_select_product_quantity [SPQ]
Choose a product [AC4]
@to_select_product [SP]
@to_ask_for_policy [AFP]
Read policies [AC3]
Search for a product [AC2]
@to_search_for_product [SFP]
Choose a product group [AC1]
@to_select_product_category [SPC]
On-lineShop
Checkout [AS14]
@to_update_stock_data [USD]
Update stocked product data [AS19]
@to_store_payment_method [STPM]
Store payment method [AS18]
@to_store_ship_method [STSM]
Store ship method [AS17]
@to_check_customer_data [CCD]
@to_store_customer_data [STCD]
Collect personal data[AS15]
Check login data[AS16]
Visualize cart [AS13]
@to_provide_cart_data [PCAD]
@to_checkout [CHKO]
Compute total [AS12]
@to_compute_price [COP]
@to_create_cart [CRC]
Remove product from cart [AS11]
@to_remove_product [RP]
Add product to cart [AS8]
Create cart [AS9]
Warn buyer [AS10]
@to_warn_customer [WC]
@to_add_product [AP]
@to_check_product_availability [CPA]
Check product quantity availability [AS7]
@to_provide_product_data [PPRD]
Show product data [AS6]
Search for a product [AS3]
Provide summarized product info [AS2]
Provide detailed product information [AS4]
@to_provide_product_data [PPRD]
Provide policy information [AS5]
@to_provide_policy_data [PPD]
@to_provide_product_data [PPRD]
@to_search_for_product [SFP]
@to_provide_product_data [PPRD]
Show the home page [AS1]

Figure 6.11: Semantically annotated on-line purchase process: for each activity and for each semantic annotation the corresponding abbreviation is reported.

on Shop's pool) and the ontology concepts used to annotate them (all reported in Figure 6.11). The latter are shown in Figure 6.14 as they are depicted by the plugin Jambalaya of the tool Protégé[12], together with their abbreviations (only is_a relationships are drawn).

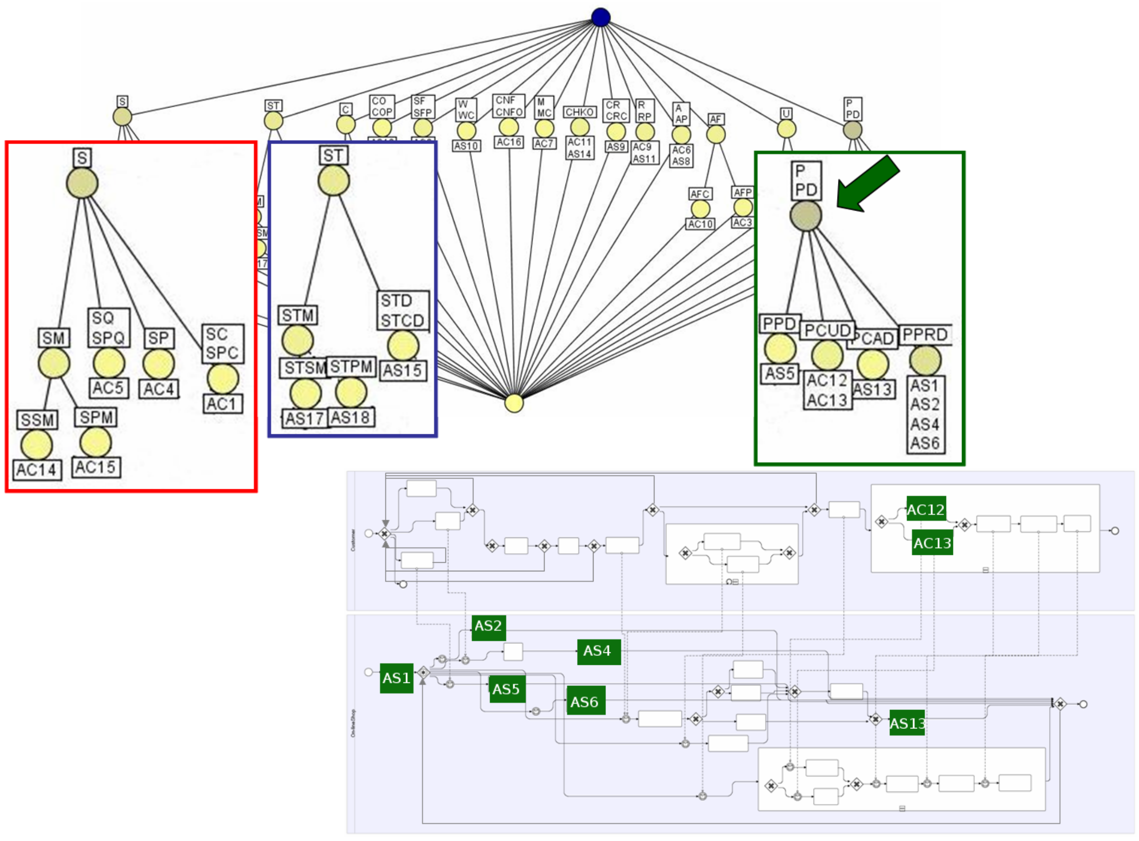


Figure 6.12: On-line purchase case study: concept lattice representing groups of process element instances sharing common semantics. Process elements in the extent of the selected FCA concept (green arrow) are highlighted at the bottom.

The FCA context for this case study relates process elements with the respective annotations. Closure of the context is automatically computed with respect to inheritance between ontology concepts. The resulting context consists of 35 BPMN elements and 50 semantic annotations (ontology concepts). We applied the tool ToscanaJ to this context and obtained the set of all its FCA-concepts arranged as a concept lattice (shown in Figure 6.12). In total, 37 FCA-concepts are obtained for this example. Sparse labelling is

[12]http://protege.stanford.edu/

used in the concept lattice in Figure 6.12. For example, the FCA-concept associated with the node marked with "SM" ("to_select_method") has the activities $AC14$ and $AC15$ in the extent (i.e., all those downward reachable). The FCA-concept labelled by the activity $AS17$ has "ST" ("to_store"), "STM" ("to_store_method") and "STSM" (abbreviation of "to_store_ship_method") as its semantic concepts (intent), i.e., all upward reachable attributes.

Concept		**|Extent|**	**Scattering**	**Judgement**
Extent	**Intent**			
{$AC12$, $AC13$, $AS1$, $AS2$, $AS4$, $AS5$, $AS6$, $AS13$}	{to_provide[P], to_provide_data[PD]}	8	22.86 %	*
{$AC1$, $AC4$, $AC5$, $AC14$, $AC15$}	{to_select[S]}	5	14.29 %	*
{$AS1$, $AS2$, $AS4$, $AS6$}	{to_provide[P], to_provide_data[PD], to_provide_product_data[$PPRD$]}	4	11.43 %	
{$AS15$, $AS17$, $AS18$}	{to_store[ST]}	3	8.57 %	*
{$AC6$, $AS8$}	{to_add[A], {to_add_product [AP]}	2	5.71 %	
{$AC3$, $AC10$}	{to_ask_for[AF]}	2	5.71 %	
{$AS7$, $AS16$}	{to_check[C]}	2	5.71 %	
{$AC11$, $AS14$}	{to_checkout[$CHKO$]}	2	5.71 %	
{$AC12$, $AC13$}	{to_provide[P], to_provide_data[PD], to_provide_customer_data[$PCUD$] }	2	5.71 %	
{$AC9$, $AS11$}	{to_remove[R], to_remove_product[RP]	2	5.71 %	
{$AC14$, $AC15$}	{to_select[S], to_select_method[SM] }	2	5.71 %	
{$AC2$, $AS3$}	{to_search_for[SF], {to_search_for_product[SFP]}	2	5.71 %	
{$AS17$, $AS18$}	{to_store[ST], {to_store_method[STM] }	2	5.71 %	
{$AC8$, $AS19$}	{to_update[U] }	2	5.71 %	
...	...	...	...	...

Table 6.4: On-line purchase case study: top of the list ranking the FCA-concepts according to their extent size. The FCA-concepts marked with an asterisk are judged as meaningful crosscutting concerns.

Table 6.4 shows the first FCA-concepts in the list of all FCA-concepts obtained for the analysed case study, ranked by decreasing level of scattering (i.e., decreasing extent size), top FCA-concept excluded. If we use a scattering threshold equal to 3 (corresponding to 8.57% of the activities in the process), we obtain the list of four candidate crosscutting concerns shown at the top of Table 6.4.

In order to evaluate the obtained result, we followed the guideline described above. By looking at the FCA-concept lattice in Figure 6.12, the three FCA-concepts labelled respectively by "to_select" ("S"), by "to_store" ("ST"), and by "to_provide" and "to_provide_data" ("P" and "PD") have been judged as meaningful crosscutting concerns, that are worth documenting explicitly. In fact, the FCA-concept labelled “S” represents all process points where the user makes some choice and expresses some preferences. The FCA-concept “ST” is associated with system’s activities devoted to storing information about the user and her current selections. The FCA-concept “P, PD” identifies all points in the workflow where information is exchanged between the user and the system. In Figure 6.12, the BPMN elements in the extent of this concept are highlighted in the process depicted at the bottom. Taken together, these three concerns convey important knowledge about user preference management, by showing when the user makes selections (“S”), what data is currently stored (“ST”) and what data is currently provided (“P, PD”). If user preference management is going to be modified and improved in the future (e.g., by adding suggestions or advertisements), knowledge about these three concerns simplifies localization of the changes, as well as consistent implementation and evaluation of their impact on the process.

Based on this analysis of the concept lattice, we can conclude that in this example our technique performed as follows: precision = 75%, recall = 100% and F-Measure = 0.86. If, instead of 3, the chosen threshold is 4, precision and recall are both 66%, and the F-Measure 0.66. The distribution of precision and recall (on the left) and the F-Measure distribution (on the right) parameterised over the threshold value are shown as solid lines in Figure 6.13.

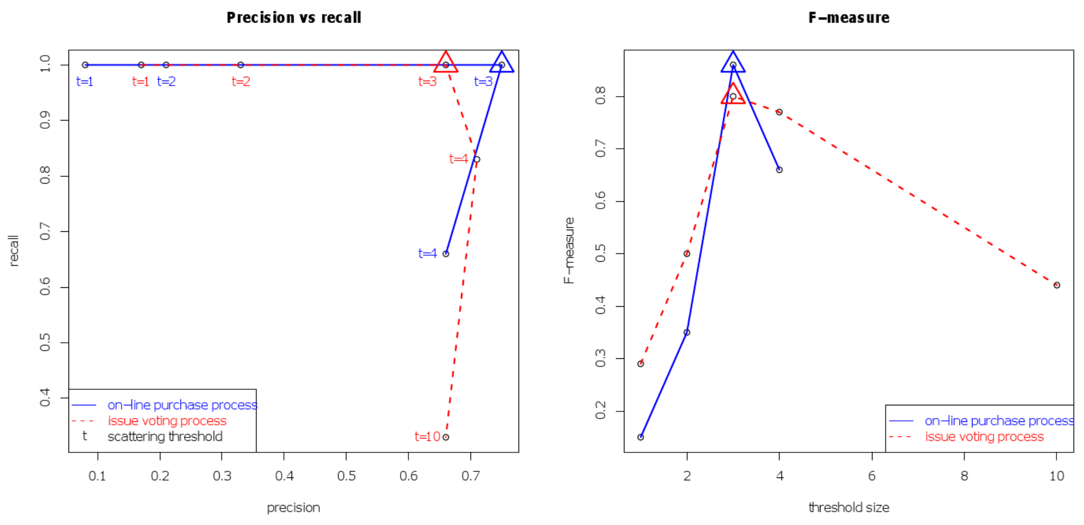


Figure 6.13: On-line purchase (solid line) and issue-voting (dashed line) case-studies: precision versus recall and F-Measure distributions. The triangles indicate the best result obtained according to the F-Measure: in both case studies the best choice for the scattering threshold (t) is t = 3.

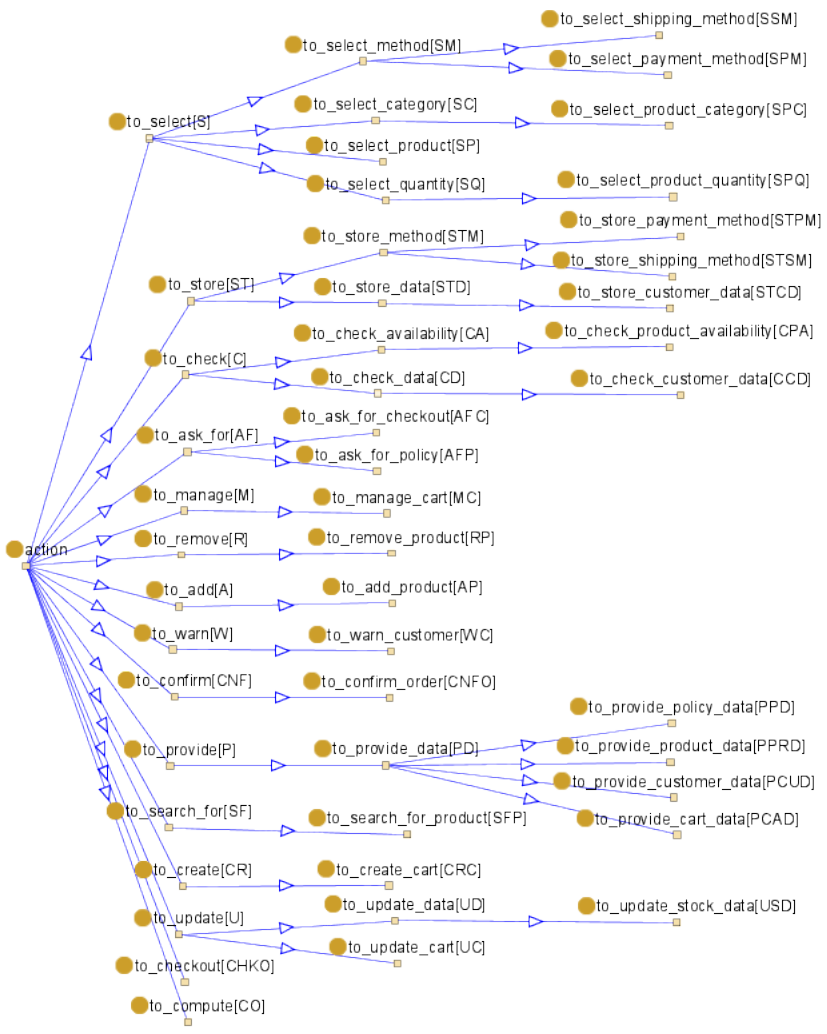


Figure 6.14: Ontology used for the annotation of the on-line purchase process. Abbreviations used to represent ontology concept names are reported next to the corresponding concept in square brackets.

E-mail Voting Case Study

The second case study analysed in this work is the process used as a working example in the BPMN 1.2 specification [135]. It describes a procedure for solving issues by means of votes provided by email. Leaving the semantics unchanged, the process structure has been slightly modified by: (i) promoting the BPD graphical objects contained in a sub-process with no label to the top level (i.e., by adding the needed gateways and removing the sub-process); and (ii) by splitting two tasks, characterized by labels that are the conjunction of different actions (e.g., “Reduce number of Voting Members and Recalculate Vote”), into a number of tasks (e.g., “Reduce number of Voting Members” and “Recalculate Vote”) ensuring one action annotation per task. The resulting process, reported in Figure 6.15, contains 14 top-level activities, 12 tasks and 2 sub-processes (including, in turn, 6 tasks each), 8 top-level gateways and 5 top-level events. It has been semantically annotated with concepts from an ontology containing 83 concepts, 21 of which have been actually used for process annotation. By analysing the concept lattice, built on the basis of the relationships between the BPMN elements and the respective annotations, and ranking the FCA-concepts according to decreasing extent size (Table 6.5), candidate crosscutting concerns are automatically detected.

As in the on-line purchase case study, a subjective evaluation has been conducted in order to assess the validity of the obtained results. By applying the same guideline, six business concerns have been identified in the issue-voting case study. Among these, one corresponds to the FCA-concept highest in the list ranked according to the scattering level, i.e., the FCA-concept labelled by "to_communicate". It represents all process points where some form of communication takes place. However, also some of its specializations have been judged as meaningful concerns, potentially requiring per se specific attention and documentation. For example, the FCA-concept labelled by "to_inform", as well as the one labelled by "to_moderate" and "to_moderate_discussion" represent the points in the workflow in which some knowledge is provided or a discussion takes place, respectively. In turn, the "to_inform" concern, can be further refined and specialized in still relevant and meaningful concerns. For example, the FCA-concepts labelled by "to_warn_about" and "to_announce" characterize specific kinds of provided information. The corresponding concerns, in fact, are useful for documenting all the points of the process generating alerts or making some information known. Separate identification and documentation of different kinds of communication, as well as different kinds of provided information, could be extremely useful, for example in all cases in which the analysts are interested in locating all the places in the process where just generic or some specific kind of communication takes place. An example of scenario of this type is when analysts have to change the management of *alerting* communications.

The last crosscutting concern that has been evaluated as a business concern corresponds to the FCA-concept labelled by "to_change" and "to_change_number". Unlike the others, the relevance of this concern is tied to the specific case-study process in which it appears, i.e., a process in which the final result about issues has to be reached by applying vote evaluation policies. Such a concern, in fact, represents all the points in the process, where the specific policies related to the vote computation are applied (e.g., the number of the proposed solutions is reduced to the most voted ones because the quorum has not been

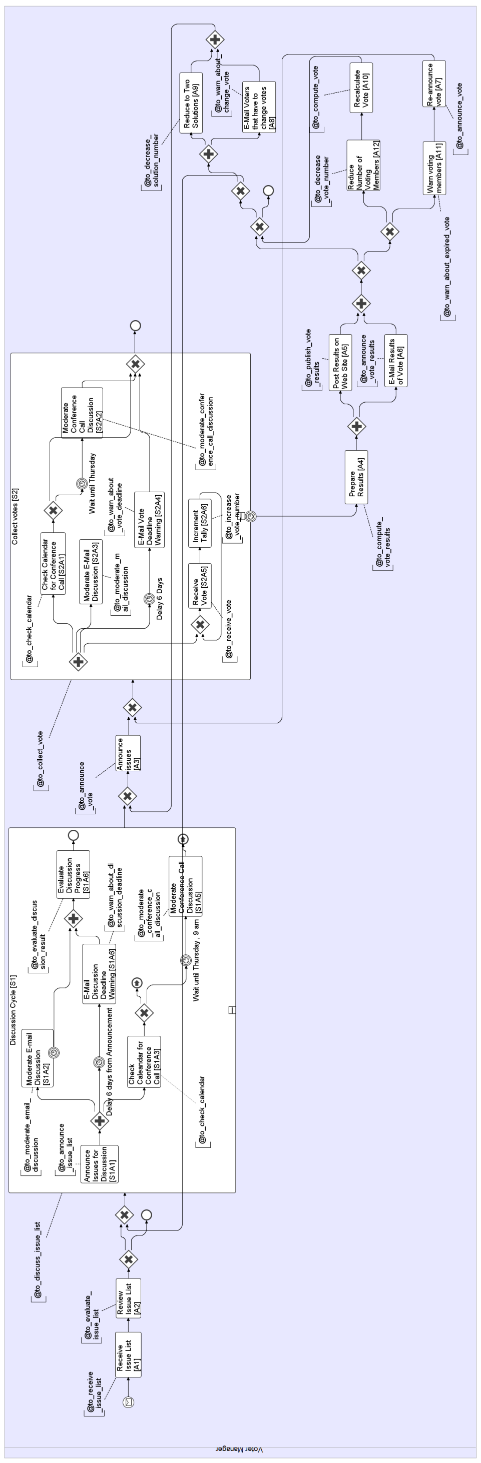


Figure 6.15: Semantically annotated e-mail vote process: for each activity the corresponding abbreviation is reported.

Concept		**|Extent|**	**Scattering**	**Judgement**
Extent	**Intent**			
$\{A3, A5, A6, A7, A8, A11, S1, S1A1, S1A2, S1A3, S1A4, S1A5, S2A1, S2A2, S2A3, S2A4\}$	{to_act }	16	61.54 %	
$\{A3, A5, A6, A7, A8, A11, S1, S1A1, S1A2, S1A4, S1A5, S2A2, S2A3, S2A4\}$	{to_act, to_communicate}	14	46.15 %	*
$\{A3, A5, A6, A7, A8, A11, S1, S1A1, S1A4, S2A4\}$	{to_act, to_communicate, to_inform}	10	38.46 %	*
$\{S1A2, S1A5, S2A2, S2A3\}$	{to_act, to_communicate, to_moderate, to_moderate_discussion}	4	15.38 %	*
$\{A8, A11, S1A4, S2A4\}$	{to_act, to_communicate, to_inform, to_warn_about}	4	15.38 %	*
$\{A3, A6, A7, S1A1\}$	{to_act, to_communicate, to_inform, to_announce}	4	15.38 %	*
$\{A2, A4, A10, S1A6\}$	{to_think }	4	15.38 %	
$\{A1, S2, S2A5\}$	{to_get }	3	11.54 %	
$\{A9, A12, S2A6\}$	{to_change, to_change_number }	3	11.54 %	*
$\{S2A4, S2A5\}$	{to_act, to_communicate to_inform, to_warn_about, to_warn_about_deadline }	2	7.69 %	
$\{A3, A7\}$	{to_act, to_communicate, to_inform, to_announce to_announce_vote }	2	7.69 %	
$\{S1A2, S2A3\}$	{to_act, to_communicate, to_moderate, to_moderate_discussion, to_moderate_email_discussion }	2	7.69 %	
$\{S1A5, S2A2\}$	{to_act, to_communicate, to_moderate, to_moderate_discussion } to_moderate_conference_call_discussion }	2	7.69 %	
$\{S1A3, S2A3\}$	{to_act, to_check , to_check_calendar }	2	7.69 %	
$\{A9, A12\}$	{to_change, to_change_number, to_decrease, to_decrease_number }	2	7.69 %	
$\{A4, A10\}$	{to_think, to_compute, to_compute_result, to_compute_vote_result}	2	7.69 %	
$\{A2, S1A6\}$	{to_think, to_evaluate}	2	7.69 %	
$\{A1, S2A5\}$	{to_get, to_receive }	2	7.69 %	
...	...	...	...	...

Table 6.5: Issue-voting case study: top of the list ranking the FCA-concepts according to their extent size. The FCA-concepts marked with an asterisk are judged as meaningful crosscutting concerns.

reached in a previous vote iteration).

The evaluation of meaningful crosscutting concerns described above allows us to compute precision, recall and F-Measure for different scattering thresholds. For example, for a scattering threshold equal to 3 (corresponding to 11,54% of the semantically annotated activities), precision and recall are respectively 66% and 100%, while the F-Measure is equal to 0.8. If instead of 3, the threshold is 4, the resulting measures are 71%, 83% and 0.77. For this second case study the two distributions (precision vs. recall and F-Measure)

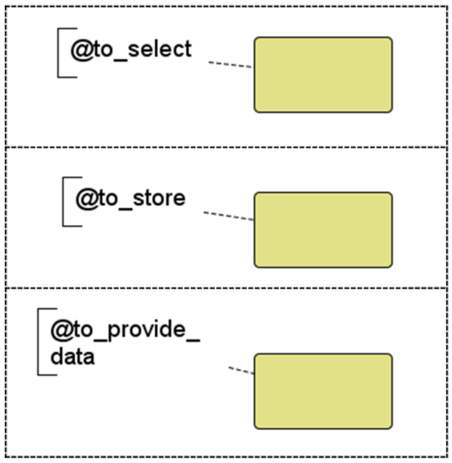


Figure 6.16: Visual query for the "user preference management" concern of the on-line purchase case study described in Subsection 6.2.2. It is the union of the three crosscutting concerns mined for the use case.

according to the scattering thresholds are depicted in Figure 6.13 as dashed lines.

6.3 Concern Documentation

Whenever business concerns are retrieved, by either querying or mining them in business processes, relevant knowledge about the synergy between the process flow and the business domain is acquired. This information can be recorded in a form that is easy to understand and visualize, and hence useful for designers and analysts to document the existence (and later the evolution) of crosscutting concerns. The graphical and intuitive nature of BPMN VQL makes it a good candidate to this purpose. Moreover, the BPMN VQL documentation query not only is available in case of process querying, but its formulation can be also easily automated in case of crosscutting concern mining.

Turning the FCA concepts that have been regarded as good candidate concerns into BPMN VQL queries, in fact, is a straightforward task: the query consists of one BPMN element and one annotation per concept. The type of the BPMN element is the least common superclass in the BPMNO, among all BPMNO-types of the elements in the FCA concept extent. The annotation is the and-composition of all annotations in the intent of the selected FCA-concept. The resulting and-expression can be simplified by replacing domain concepts in the and-expression, that are hierarchically structured in the BDO, with their least common superclass. For example, the concept labelled "P, PD" in the concept lattice of the on-line purchase case study (Figure 6.12) has only tasks in the extent. Hence, the BPMNO-type of the BPMN element in the query is a task. The and-expression composing the semantic annotations of this FCA-concept is "@to_provide $\wedge$ @to_provide_data", which can be automatically simplified into "@to_provide_data", based on the inheritance relationship between PD and its superconcept P in the domain ontology. Finally, when multiple FCA-concepts are involved, the associated BPMN elements (semantically annotated with the intent concepts) become alternatives of the documentation query (i.e., they are composed by means of the BPMN VQL OR operator).

The query documenting the "user preference management" concern of the on-line purchase case study described in Subsection 6.2.2 (Figure 6.12) is shown in Figure 6.16 as the union (alternative) of the three crosscutting concerns that have been automatically mined.

Chapter 7

Business Process Aspectization

"A designer knows he has achieved perfection
not when there is nothing left to add,
but when there is nothing left to take away."
Antoine de Saint-Exupéry

Though allowing to represent classic process perspectives, existing process modelling languages do not provide any constructs to describe crosscutting concerns. In Chapter 6 we propose a manual and a semi-automatic approach for retrieving crosscutting concerns in business processes and to separately document them, thus supporting business designers and analysts in comprehension and analysis tasks. However, modularization and separate management (specification and evolution, not just documentation) of crosscutting concerns would be helpful for design and refactoring purposes.

In the literature, *aspects* are proposed as a possibility to cope with the separation of concerns for general purpose programming languages [97]. In business process modelling, aspects would allow designers to modularize information in separate views, hence easing the job of modelling and maintaining crosscutting concerns, but also of reading and understanding process models by providing a view of the process that is oblivious of crosscutting concerns.

The separate modularization of crosscutting concerns would allow analysts to deal with business process modelling issues, as for example exception handling. In fact, caring about exceptional behaviours and verifying their correct management is a key factor for process model robustness [43]. However, the management of exceptional flows can introduce a high complexity in processes, thus business designers often prefer to focus only on the main flow (i.e., the so called "happy path"). The use of aspects could hence allow to manage exceptional behaviours while preserving the readability of the happy path.

In this chapter we first present an approach for the modularization of business process concerns into aspects (Section 7.1), we then provide an example of use of the proposed approach, by applying it to exception handling (Section 7.2) and we finally provide a preliminary evaluation of the applicability of the approach (Section 7.3).

7.1 Semantically enhanced aspects

Process description languages do not provide any mechanism to modularize crosscutting concerns, i.e., concerns that, being scattered across the process, go beyond the local boundary of sub-process elements. On the other hand, process models, in order to guarantee readability and understandability, would need mechanisms for modularizing special structural and domain concerns.

We consider the use of *aspects* [58] to cope with this lacking capacity of process languages at design time. An *aspect* is a module that encapsulates a secondary behaviour of a main view. Taking advantage of the separation of concerns, designers can deal with aspects separately and independently from the main view. If needed, aspects can be added to the principal perspective in the weaving phase, when the "woven" (integrated) process is generated, thus providing a global view of the process. This allows the business experts to manipulate each crosscutting concern locally, leaving the weaver the responsibility of propagating the changes consistently and completely to all process portions matching the pattern change to be applied. Moreover, while aspects are mainly used for capturing non-functional requirements, in case of processes enriched with semantic annotations, the semantic domain-related properties of crosscutting concerns can also be captured.

As classic aspect-based languages (e.g., AspectJ [97]) are tailored to the specific textual programming language they enhance with aspects, aspect-based languages for processes should extend the process language they want to aspectize. This would also have the advantage to make it easier for language experts learning the aspect-based extension. Some of the aspect-based languages for processes proposed in the literature, in fact, partially realize this idea. For instance, AO4BPEL [7] is an aspect-based extension for BPEL [39] executable processes; its syntax allows to describe the new behaviour to be added or removed in the form of BPEL fragments. Similarly, in case of (semantically annotated) BPMN process models, in order to support business designers, analysts and managers in the use of aspects for the modularization of concerns crosscutting the processes, we propose an aspect-based language based on BPMN with semantic annotations, the BPMN VRL (BPMN Visual Rule Language).

In this section we first provide some details about Aspect Oriented Programming and aspects (Subsection 7.1.1) and we then present the BPMN VRL language we propose for the aspect definition of BPMN semantically annotated processes (Subsection 7.1.2).

7.1.1 Aspect Oriented Programming

Aspect Oriented Programming (AOP) [58] investigates the separation of crosscutting concerns and their modularization into aspects. An aspect is a module containing information about the three main dimensions ("where", "what" and "when") of a given crosscutting concern. A so called *pointcut* designator answers the "where?" question by providing a condition to specify a set of so called *join points*, i.e., precise points in the execution flow that the aspect intercepts, by means of some quantification mechanism. An *advice*, instead, answers the "when?" and "what?" questions, by specifying how to realize the concern and when in the join points of interest (i.e., before, after or around) the concern execution has to be activated. AOP is therefore *quantification* (through the conditions that filter the execution flow) and *obliviousness* (the primary program ignores, i.e., has no reference to, advices) [58]. The aspect is eventually *woven*, i.e., integrated with the core functionality, at compile time (*static weaving*) or at runtime (*dynamic weaving*). Classic examples of crosscutting concerns that can be modularized into aspects are related to non-functional properties, such as logging and transactional functionalities. Both of them are scattered across different primary modules and tangled with other concerns. The idea is therefore to extract and modularize them, making the principal code oblivious of logging and transaction concerns.

In business processes an aspect is a separate module that adds behaviour to the principal decomposition of the process by specifying where the behaviour has to be added ("where?") and what kind of behaviour has to be added ("what?"). In detail the pointcut designator answers the "where?" question, by providing a condition that allows to intercept a set of precise points (join points) in the process execution flow (quantification). An example of pointcut in BPMN processes, could be a sub-process, thus indicating that the new behaviour has to be added at each sub-process occurrence. The advice, instead, answers the "what?" question by specifying how to realize the concern. An example of advice for BPMN processes could be an intermediate error event directly followed by an end event, to be added on the boundary of the specific sub-process (pointcut). The main view of the process is hence oblivious of (i.e., has no reference to) aspects, that are woven in the core functionality only when needed [58]. This allows the business experts to manipulate each crosscutting concern locally, leaving the weaver the responsibility of propagating the changes consistently and completely to all process portions that match the pattern change to be applied. For instance, for the aspect described above, the weaver will enrich each sub-process in the process model with an exception handler terminating the sub-process whenever an exception occurs (i.e., with the error event on the sub-process boundary and with the connected end event).

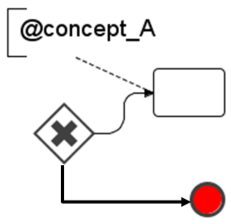
7.1.2 BPMN VRL

The purpose of BPMN VRL is to provide business experts with an intuitive and easy-to-learn means to modularize crosscutting concerns into aspects in semantically annotated business processes. This would allow managing crosscutting concerns separately from the main view and, when necessary, integrating them into the main flow of the process by exploiting the weaving mechanism.

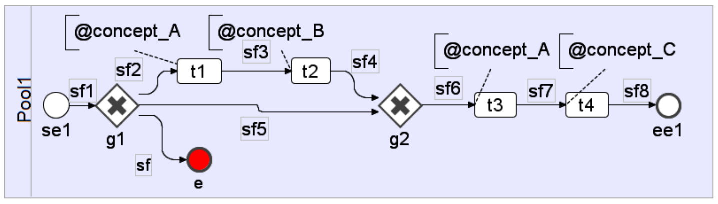
Similarly to BPMN VQL (described in Chapter 6), BPMN VRL has been designed so as to be close to business experts' knowledge. It is hence a graphical language extending the BPMN. It is a rule language that exploits the same mechanism of the BPMN VQL (see Subsection 6.1.1 of Chapter 6) for the quantification ("where?") and that, additionally, provides a process manipulation mechanism for the process updates ("what?").

Each BPMN VRL rule is expressed in a visual language which consists of two parts: the "matching" and the "update" part. The first part looks like a BPMN VQL query. It is composed of a *matching pattern* (corresponding to the BPMN VQL matching criterion), which represents the pattern to be matched (in terms of graph matching and domain semantics) and of a *selection sub-pattern* (corresponding to the BPMN VQL selection pattern), which is the subset of matching pattern components, whose occurrences are returned to the user. In BPMN VRL, however, the selection sub-pattern is not explicitly represented with a darker background (as in BPMN VQL queries), but it is inferred from the "update" part of the rule. In detail, the selection sub-pattern is the subset of matching pattern components directly involved in the modification (i.e., the BPMN elements connected to new BPMN elements to be added or BPMN elements to be removed, respectively). The "update" part, with a darker background, thicker lines and bold font style, represents the modifications (behaviour addition or removal) to apply whenever a match occurs. The addition of new behaviour to the process is represented by using BPMN elements (with a darker background and thicker lines), while the REMOVE operator is introduced for representing the removal of one or more BPMN elements. The REMOVE operator is depicted as a filled cross, over the BPMN elements to be removed.

Figure 7.1a shows an example of a BPMN VRL rule that inserts an end event as new

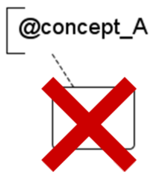


(a)

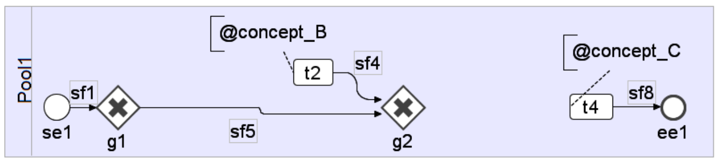


(b)

Figure 7.1: Simple example of BPMN VRL rule adding behaviour



(a)



(b)

Figure 7.2: Simple example of BPMN VRL rule removing behaviour

alternative for a data-based exclusive gateway directly preceding a task of BDO-type concept_A. The BPMN elements with white background and thinner lines (i.e., the concept_A task, the data-based XOR gateway and their connecting sequence flow) represent the pattern to be matched against the process. The BPMN elements with darker background and thicker lines (i.e., the end event and the sequence flow connecting it to the gateway) represent instead the behaviour to be added to the process, whenever the pattern matches.

Figure 7.2a shows an example of a BPMN VRL rule that removes tasks of BDO-type concept_A from the process. The BPMN elements with white background and thinner lines (i.e., the concept_A task) represent the pattern to be matched against the process. The BPMN elements crossed by the REMOVE operator (i.e., the same concept_A task) are the elements to be removed.

Once a BPMN VRL rule has been defined by a business expert, in the rule weaving phase the changes described in the "update" part of the rule are automatically applied to the process (i.e., to the knowledge base and hence to the BPD), by changing all occurrences in the process satisfying the matching criterion. Operationally: (i) the occurrences of the semantically annotated process satisfying the matching criterion and specified by the selection sub-pattern are identified by querying the knowledge base encoding the process (the BPKB); (ii) for each retrieved occurrence, the modifications specified in the "update" part of the rule are applied, by adding/removing instances to/from the BPKB.

In detail, the BPMN VRL aspect weaving is realized in two steps: (i) the "matching" part of the rule is translated into a SPARQL query (similarly to the *matching criterion* of BPMN VQL queries described in Chapter 6); (ii) the "update" part is translated into a set of assertions affecting the BPKB.

The "update" part of the rule builds upon each set of instances resulting from the query execution[1]. For each set of instances returned by the query: (a) a set of new BPM-type, BPM-structural, BPM-semantic assertions is added to the BPKB according to the elements with darker background and thicker lines in the rule; (b) a set of assertions in the BPKB is removed. The assertions to be eliminated are all those involving BPD instances associated to elements to be removed according to the BPMN VRL rule (i.e., all those crossed by the REMOVE operator) and, in case of flow objects, also all assertions involving BPMN connecting objects (i.e., sequence flows, message flows and associations) and text annotations[2] "pending" in the process (i.e., connecting objects without source or target or text annotation not associated to any flow object) due to the elimination of the flow object. To this purpose, a new query that allows to identify all instances of "pending" connecting objects and text annotations to be removed, is formulated. However, the query does not guarantee the reachability of all the flow objects in the process flow, as well as of data objects associated to one or more activities. Caring about a correct use of the REMOVE operator so as to guarantee the reachability of all the process elements is left to the aspect designer. Support tools could be of course developed to help the designer handle any reachability problem in the woven process. For a generic flow object *fo* to be removed according to a BPMN VRL rule, the query (7.1) can be executed for collecting the "pending" BPD elements to be eliminated.

[1]BPMN VRL rules are not recursive, i.e., the pattern is matched against the non-woven process and modifications are applied only to the result of the match (and not to partially updated parts of the process).

[2]Special care is reserved to text annotations in the BPMN VRL language due to their large use in semantically annotated processes.

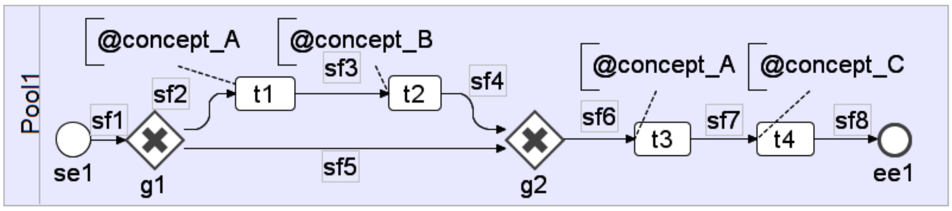


Figure 7.3: Example process

```
SELECT ?co, ?do
WHERE {
        ?co rdf:type BPMNO:connecting_object
        {?co bpmn:has_connecting_object_source_ref fo.}
        UNION
        {?co bpmn:has_connecting_object_target_ref fo.}                    (7.1)
        OPTIONAL
        {?co rdf:type BPMNO:association.
        ?do rdf:type BPMNO:annotation.
        ?co bpmn:has_connecting_object_source_ref ?do.
        ?co bpmn:has_connecting_object_target_ref fo.}
      }
```

For example, weaving the aspect in Figure 7.1a on the example process in Figure 7.3, requires as first step the execution of the following SPARQL query:

```
SELECT ?g1
WHERE {
        ?t1rdf:typeBPMNO:task.
        ?t1rdf:typeBDO:concept_A.
        ?g1rdf:typeBPMNO:data_based_exclusive_gateway.                      (7.2)
        ?sf1BPMNO:has_sequence_flow_source_ref?g1.
        ?sf1BPMNO:has_sequence_flow_target_ref?t1.
        ?sf1rdf:typeBPMNO:sequence_flow.
      }
```

It retrieves all the instances of data-based exclusive gateways directly followed by at least a concept_A task, as shown in Figure 7.1b. By running the query on the process in Figure 7.3, we only get as result $g1$. The update part of the rule is hence applied to the only result of the query. In detail, two new BPM-type assertions are added to the BPKB: end_event(e) and sequence_flow(sf), where e and sf represent the new end event and the new sequence flow, respectively. Moreover, two BPM-structural assertions are also added: has_sequence_flow_source_ref $(sf, g1)$ and has_sequence_flow_target_ref (sf, e).

When weaving the aspect in Figure 7.2a on the same process, instead, the following simpler SPARQL query is executed:

$$
\begin{aligned}
&\mathsf{SELECT}\ ?t\\
&\mathsf{WHERE}\ \{\\
&\qquad ?t\mathsf{rdf{:}typeBPMNO{:}task.}\\
&\qquad ?t\mathsf{rdf{:}typeBDO{:}concept\_A.}\\
&\qquad \}
\end{aligned}
\tag{7.3}
$$

In this case, all the instances of concept_A tasks are retrieved. By running the query on the process in Figure 7.3, we get two results: $t1$ and $t3$. The update part of the rule is hence applied twice: once to $t1$ and the other to $t3$. For each of them not only all the BPKB assertions involving the considered task have to be removed, but, since each result is a flow object, all the assertions involving one of its connecting objects or text annotations are also removed. In practice, in case of $t1$, the BPM-type assertion task($t1$), the BPM-semantic assertion concept_A($t1$), the BPM-structural assertions has_sequence_flow_target_ref $(sf2, t1)$, has_sequence_flow_source_ref $(sf3, t1)$, has_connecting _object_target_ref $(as1, t1)$ (assuming that $as1$ is the name of the association connecting the task $t1$ to its textual annotation "@concept_A") are removed. Moreover, the execution of the query (7.1) returns three result sets: $sf2$, $sf3$ and ($as1$, $an1$), where we assume that $an1$ is the name of $t1$'s textual annotation "@concept_A", thus leading to the deletion of four BPM-type assertions (sequence_flow($sf2$), sequence_flow($sf3$), association($as1$) and annotation($an1$)) and of the BPM-structural assertion has_connecting_ object_target_ref $(as1, an1)$. Similarly, in case of $t3$, the following assertions will be removed: task($t3$), concept_A($t3$), has_sequence_flow_target_ref $(sf6, t3)$, has_sequence_flow_ source_ref $(sf7, t3)$, has_connecting_object_target_ref $(as3, t3)$, sequence_flow($sf6$), sequence_flow($sf7$), association ($as3$), annotation($an3$) and has_connecting_object_target_ref $(as3, an3)$, where we assume that $an3$ is the name of $t3$'s "@concept_A" annotation and $as3$ the name of the association connecting $an3$ to $t3$. This last BPMN VRL rule (the process resulting from its application is shown in Figure 7.2b) is an example of aspect designed without caring about the flow of the woven process since, after the weaving, it leaves parts of the process not connected.

In the following we provide some examples of BPMN VRL rules describing aspects applied to the semantically annotated process reported in Figure 7.4 (the same as used in Chapter 5). In detail we describe the BPMN VRL aspect, the SPARQL translation of the matching pattern, the assertions to be added, as well as the instances whose assertions have to be removed for the "update" part of the rule. Finally, a view of the process

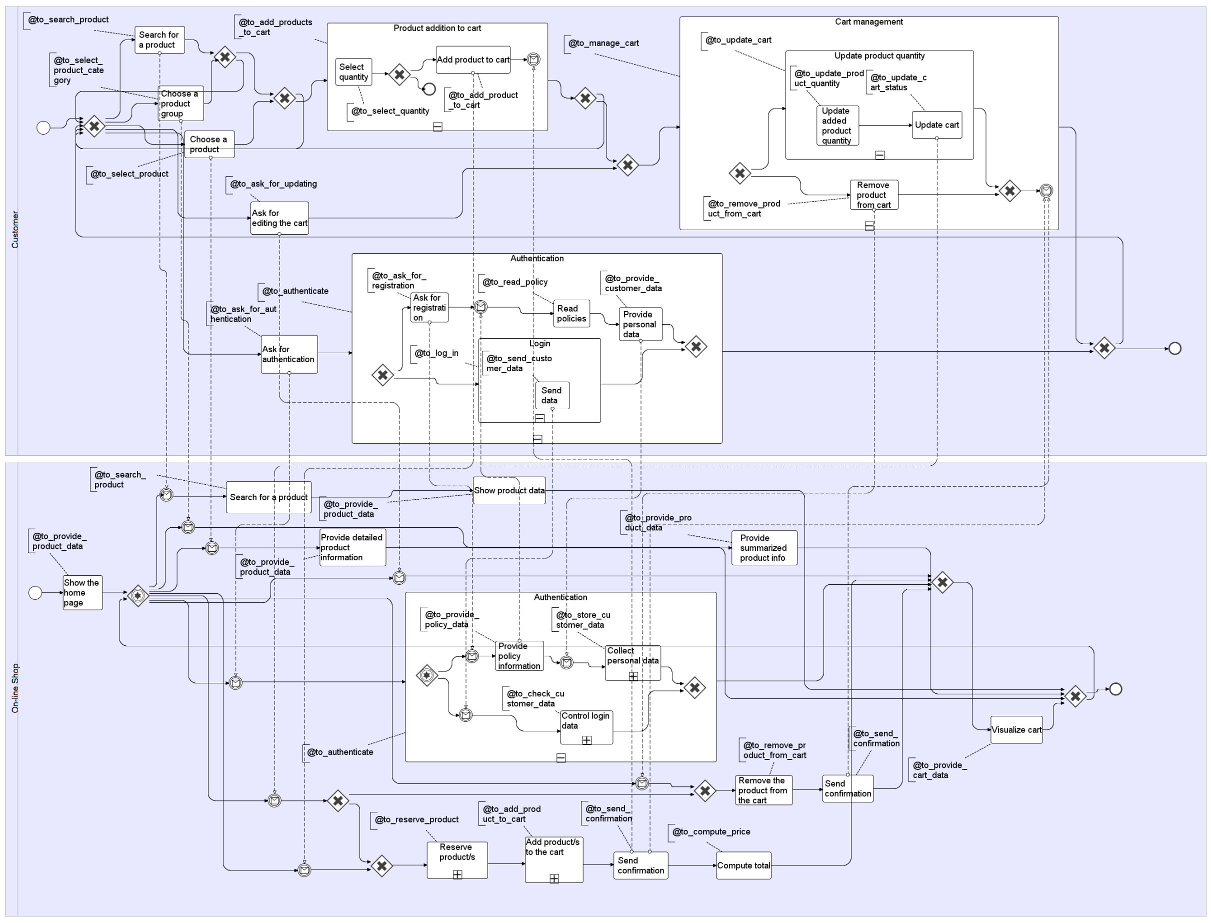


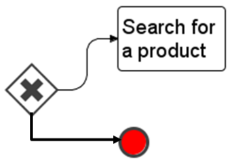
Figure 7.4: A portion of the On-line shopping business process diagram.

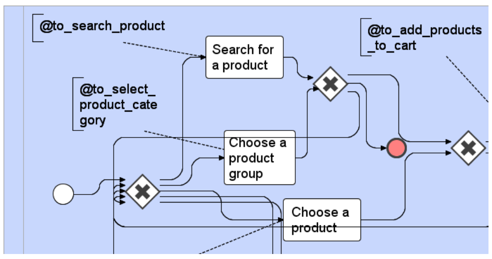
obtained after the aspect weaving[3] is also reported.

Aspects adding behaviour. Figure 7.5 shows an example of a BPMN VRL rule that, in the weaving phase, adds an end event (and hence also the sequence flow required for the connection) outgoing from each data-based XOR gateway having as outgoing alternative an activity labelled with "Search for a product". The BPMN elements with white background and thinner lines (i.e., the data-based XOR gateway and the task labelled with "Search for a product" connected by at least a sequence flow and the sequence flow itself) represent the pattern to be matched against the process. The BPMN elements with darker background and thicker lines (i.e., the end event and the sequence flow connecting it to the gateway) represent instead the behaviour to be added to the process, whenever the pattern matches. Finally, the selection sub-pattern contains the only BPMN element

[3]In order to ease reading of the example, only views of the updated process in the Customer pool are reported and query prefixes are omitted.

involved in the modification, i.e., the data-based exclusive gateway. The variable representing the gateway is, in fact, the only variable in the select clause of the SPARQL query. For each solution *g1* of the query, the two BPM-type and the two BPM-structure assertions reported in Figure 7.5 are added to the BPKB[4].





SPARQL Query:
```
SELECT ?g1
WHERE {
  ?t1 rdf:type bpmn:task.
  ?t1 bpmn:has_flow_object_name "Search for a product".
  ?g1 rdf:type bpmn:data_based_exclusive_gateway.
  ?sf1 bpmn:has_sequence_flow_source_ref ?g1.
  ?sf1 bpmn:has_sequence_flow_target_ref ?t1.
  ?sf1 rdf:type bpmn:sequence_flow.
}
```

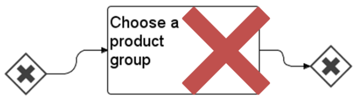
For each result $g1$:
ASSERTIONS TO BE ADDED:
end_event(e)
sequence_flow(sf)
has_sequence_flow_source_ref($sf, g1$)
has_sequence_flow_target_ref(sf, e)

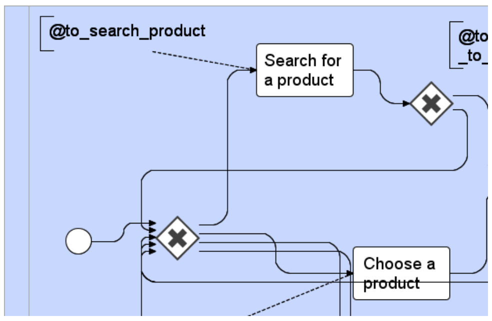
Figure 7.5: Example of a BPMN VRL rule adding behaviour: in the weaving phase, an outgoing end event is added to each data-based XOR gateway having as outgoing alternative at least a task labelled with "Search for a product".

Aspects removing behaviour. Figure 7.6 shows an example of a BPMN VRL rule that, in the weaving phase, removes all tasks labelled with "Choose a product group" and connected to two data-based gateways. The BPMN elements with white background and thinner lines (i.e., the two data-based XOR gateway, the task labelled with "Choose a product group" and directly connected to them, inbound and outbound, respectively, as well as the corresponding sequence flows) represent the pattern to be matched against the process. The task labelled with "Choose a product group" crossed by the REMOVE operator represents the behaviour to be removed from the process, whenever the pattern matches. Finally, the selection sub-pattern is composed of the BPMN elements involved in the modification: in this case, the only task crossed by the REMOVE operator. The select clause of the SPARQL query, hence, contains the variable *?t1* representing the task "Choose a product group". For each solution $t1$ of the query, all the BPM-type, BPM-structural and BPM-semantic assertions involving $t1$ will be removed. Moreover,

[4]For each distinct query result set $g1$, e and sf represent new instances.

since the removed object is a flow object, all the assertions involving connecting objects and text annotations related to the removed flow object are also eliminated (i.e., for each result $t1$ of the query reported in Figure 7.6, all the assertions involving the results of the query (7.1) with $fo = t1$ are removed)[5].





SPARQL Query:

```
SELECT ?t1
WHERE {
  ?t1 rdf:type bpmn:task.
  ?t1 bpmn:has_flow_object_name "Choose a product group".
  ?g1 rdf:type bpmn:data_based_exclusive_gateway.
  ?sf1 bpmn:has_sequence_flow_source_ref ?g1.
  ?sf1 bpmn:has_sequence_flow_target_ref ?t1.
  ?sf1 rdf:type bpmn:sequence_flow.
  ?g2 rdf:type bpmn:data_based_exclusive_gateway.
  ?sf2 bpmn:has_sequence_flow_source_ref ?t1.
  ?sf2 bpmn:has_sequence_flow_target_ref ?g2.
  ?sf2 rdf:type bpmn:sequence_flow.
}
```

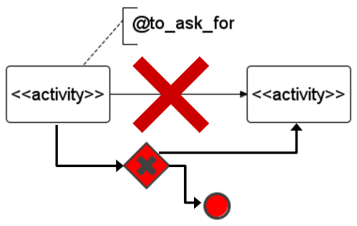
For each result $t1$:
INSTANCES TO BE REMOVED:
$t1$
$sf1$
$sf2$
as
an

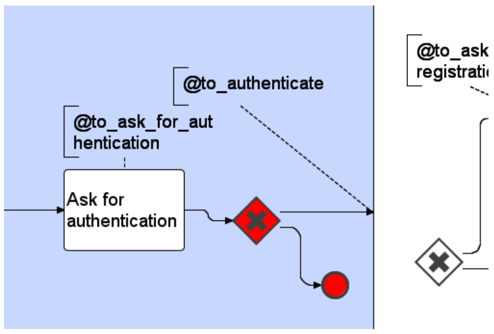
Figure 7.6: Example of a BPMN VRL rule removing behaviour: in the weaving phase, all tasks labelled with "Choose a product group" and connected, inbound and outbound, to event-based gateways are removed.

Aspects adding and removing behaviour. Figure 7.7 shows an example of a BPMN VRL rule that, in the weaving phase, adds the end event after the to_ask_for activities directly followed by another activity. In detail, it removes the existing sequence flow connecting the two activities and adds a data-based XOR gateway with two outgoing sequence flows, one having as target the activity directly following the to_ask_for activity in the original process and, the other, connected to a new end event. The matching pattern is provided by the BPMN elements with white background and thinner lines, i.e., the two activities (both tasks and sub-processes, since the "<< activity >>" stereotype has been

[5]For each result set $t1$, $sf1$ and $sf2$ represent the sequence flows connecting the first gateway to $t1$ and $t1$ to the second gateway, respectively. an and as are the textual annotation and the corresponding association used for annotating $t1$ with the domain information. All assertions involving each of these instances are removed.

used for representing the BPMNO activity hierarchy) and their connecting sequence flow represent the pattern to be matched against the process. The BPMN elements with darker background and thicker lines (i.e., the data-based exclusive gateway, the end event and the three sequence flows), represent, instead, the behaviour to be added to the process. Finally, the sequence flow crossed by the REMOVE operator represents the behaviour to be removed from the process, whenever the pattern matches. The selection sub-pattern is composed, in this case, of the two activities and of their connecting sequence flow. The select clause of the SPARQL query contains, hence, the variable *?a1* representing the to_ask_for activity, the variable *?a2* denoting the second activity and *?sf1* representing their connecting sequence flow. For each solution ($a1$, $a2$, $sf1$), the five BPM-type and the six BPM-structural assertions reported in Figure 7.7 are added to the BPKB[6] and all the BPM-type, BPM-structural and BPM-semantic assertions involving $sf1$ are removed.





SPARQL Queries:

```
SELECT ?a1 , ?a2 , ?sf1
WHERE {
  ?a1 rdf:type bpmn:activity.
  ?a1 rdf:type bdo:to_ask_for.
  ?a2 rdf:type bpmn:activity.
  ?sf1 bpmn:has_sequence_flow_source_ref ?a1.
  ?sf1 bpmn:has_sequence_flow_target_ref ?a2.
  ?sf1 rdf:type bpmn:sequence_flow.
}
```

For each result ($a1$, $a2$, $sf1$)
ASSERTIONS TO BE ADDED:
data_based_exclusive_gateway(g)
end_event(e)
sequence_flow($sf2$)
has_sequence_flow_source_ref($sf2$, $a1$)
has_sequence_flow_target_ref($sf2$, g)
sequence_flow($sf3$)
has_sequence_flow_source_ref($sf3$, g)
has_sequence_flow_target_ref($sf3$, $a2$)
sequence_flow($sf4$)
has_sequence_flow_source_ref($sf4$, g)
has_sequence_flow_target_ref($sf4$, e)
INSTANCES TO BE REMOVED:
$sf1$

Figure 7.7: Example of a BPMN VRL rule adding and removing behaviour: after the weaving phase, it will be possible to end the current (sub-)process starting from all the to_ask_for activities directly followed by another activity.

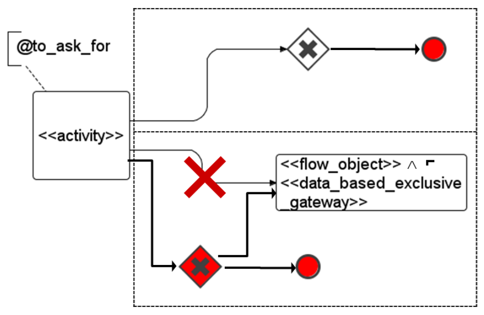
Aspects using the BPMN VQL OR operator. Figure 7.8 shows an example of

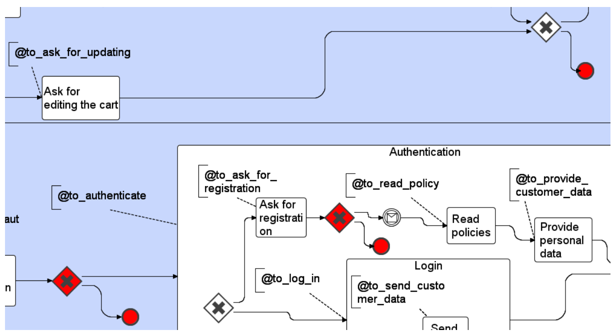
[6]For each distinct query result set ($a1$, $a2$, $sf1$), g, e, $sf2$, $sf3$ and $sf4$ represent new instances.

a BPMN VRL rule that, in the weaving phase, adds the end event after the to_ask_for activities. In detail, if the to_ask_for activity is followed by a data-based XOR gateway, the rule only adds an end event to the gateway alternatives; otherwise (i.e., if the activity is followed by a flow object that is not a data-based exclusive gateway), the rule removes the existing sequence flow connecting the activity with the flow object and adds a data-based XOR gateway with two outgoing sequence flows, one having as target the flow object directly following the to_ask_for activity in the original process and, the second, connected to a new end event. It uses the BPMN VQL OR operator, thus allowing the matching of more than one pattern. The BPMN elements with white background and thinner lines, i.e., the to_ask_for activity and either the directly connected data-based exclusive gateway or the directly connected flow object (as well as the connecting sequence flows) represent the pattern to be matched against the process. The behaviour to be added to the process is provided, instead, by the BPMN elements with darker background and thicker lines (i.e., either the end event and the connecting sequence flow, in the first OR alternative, or the data-based exclusive gateway, the end event and the three sequence flows in the second OR alternative). Finally, the sequence flow crossed by the REMOVE operator represents the behaviour to be removed from the process, whenever the second of the OR alternatives matches (i.e., when the to_ask_for activity is not directly followed by any data-based exclusive gateway). The selection sub-pattern is composed of the to_ask_for activity, the data-based XOR gateway, the flow object directly following the activity and the sequence flow connecting the activity with the flow object. The select clause of the query contains, hence, the variable *?a1* representing the to_ask_for activity, the variable *?g1* representing the gateway, when the first pattern matches, and *?a2* and *?sf2* representing the flow object and the sequence flow, when the second OR alternative matches. For each result of the query, either the two BPM-type and the three BPM-structural assertions (in case $g1$ exists), or the five BPM-type and the six BPM-structural assertions (in case $a2$ exists) of Figure 7.8 are added to the BPKB[7]. Moreover, all the BPM-type, BPM-structural and BPM-semantic assertions involving $sf2$, if this is valued, are removed.

Aspects using the BPMN VQL NOT operator. Figure 7.9 shows an example of a BPMN VRL rule that, in the weaving phase, adds the end event after the to_remove activities, if they are followed by gateways. In detail, the new end event is added only if the gateway (following the to_remove activity) does not already have an outgoing edge with an end event as target. It uses the BPMN VQL NOT operator, thus allowing to discard

[7]For each distinct query result set ($a1$, $g1$, $a2$, $sf1$), e and $sf4$ or g, e, $sf4$, $sf5$ and $sf6$ represent new instances if $g1$ or $a2$ is valued, respectively.





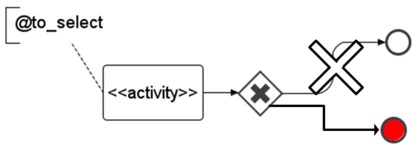
SPARQL Queries:

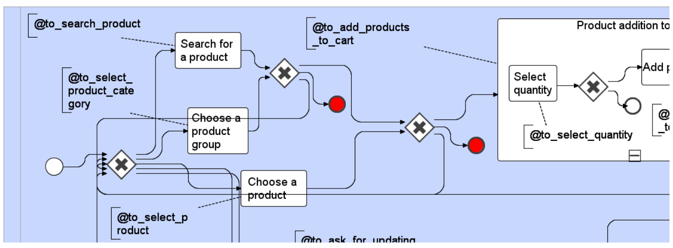
SELECT *?a1* , *?g1*, *?a2* , *?sf2*
WHERE {
  ?a1 rdf:type bpmn:activity.
  ?a1 rdf:type bdo:to_ask_for.
  { *?sf1* bpmn:has_sequence_flow_source_ref *?a1*.
   ?sf1 bpmn:has_sequence_flow_target_ref *?g1*. }
   ?g1 rdf:type bpmn:data_based_exclusive_gateway.
   ?sf1 rdf:type bpmn:sequence_flow }.
UNION
  { *?sf2* bpmn:has_sequence_flow_source_ref *?a1*.
   ?sf2 bpmn:has_sequence_flow_target_ref *?a2*.
   ?sf2 rdf:type bpmn:sequence_flow }.
   ?a2 rdf:type bpmn:flow_object.
   ?sf3 bpmn:has_sequence_flow_target_ref *?a1*.
   ?sf3 bpmn:has_sequence_flow_target_ref *?a3*.
   ?sf3 rdf:type bpmn:sequence_flow }.
   ?a3 rdf:type bpmn:data_based_exclusive_gateway.
   FILTER (*?a2*!= *?a3*) }
}

For each result ($a1$, $g1$, $a2$, $sf2$):
ASSERTIONS TO BE ADDED:
if $g1$ is valued
 end_event(e)
 sequence_flow($sf4$)
 has_sequence_flow_source_ref($sf4$, $g1$)
 has_sequence_flow_target_ref($sf4$, e)
if $a2$ is valued
 data_based_exclusive_gateway(g)
 end_event(e)
 sequence_flow($sf4$)
 has_sequence_flow_source_ref($sf4$, $a1$)
 has_sequence_flow_target_ref($sf4$, g)
 sequence_flow($sf5$)
 has_sequence_flow_source_ref($sf5$, g)
 has_sequence_flow_target_ref($sf5$, $a2$)
 sequence_flow($sf6$)
 has_sequence_flow_source_ref($sf6$, g)
 has_sequence_flow_target_ref($sf6$, e)
INSTANCES TO BE REMOVED:
if $sf2$ is valued
 $sf2$

Figure 7.8: Example of a BPMN VRL rule using the BPMN VQL OR operator: after the weaving phase, it will be possible to end the current (sub-)process from all the to_ask_for activities.

specific patterns in the matching. The matching pattern is provided by the to_select activity, the directly connected data-based exclusive gateway and the negated outgoing end event. The BPMN elements with darker background and thicker lines (i.e., the end event and the connecting sequence flow), represent, instead, the behaviour to be added to the process. The selection sub-pattern is composed of the data-based exclusive gateway. The select clause of the query contains, hence, the variable *?g1* representing the data-based exclusive gateway. For each result of the query, the two BPM-type and the three BPM-structural assertions reported in Figure 7.9 are added to the BPKB[8].





SPARQL Queries:

```
SELECT ?g1
WHERE {
  ?a1 rdf:type bpmn:activity.
  ?a1 rdf:type bdo:to_select.
  ?sf1 bpmn:has_sequence_flow_source_ref ?a1.
  ?sf1 bpmn:has_sequence_flow_target_ref ?g1. }
  ?g1 rdf:type bpmn:data_based_exclusive_gateway.
  ?sf1 rdf:type bpmn:sequence_flow }.
  FILTER (
   NOT EXISTS {
    ?e1 rdf:type bpmn:end_event.
    ?sf2 bpmn:has_sequence_flow_source_ref ?g1.
    ?sf2 bpmn:has_sequence_flow_target_ref ?e1.
    ?sf2 rdf:type bpmn:sequence_flow
   }.)
}
```

For each result $g1$:
ASSERTIONS TO BE ADDED:
end_event(e)
sequence_flow($sf3$)
has_sequence_flow_source_ref($sf3, g1$)
has_sequence_flow_target_ref($sf3, e$)

Figure 7.9: Example of a BPMN VRL rule using the BPMN VQL NOT operator: after the weaving phase, it will be possible to end the current (sub-)process from all the to_select activities directly followed by a gateway.

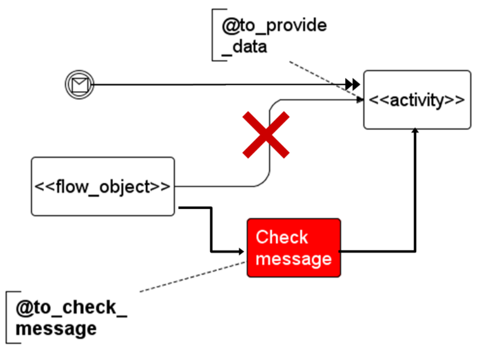
Aspects using the BPMN VQL PATH operator. Figure 7.10 shows an example of a BPMN VRL rule that, in the weaving phase, adds a request check immediately before activities providing data and preceded by the receipt of a message. In detail, a new to_check_message task is added immediately before any to_provide_data activity, for which at least a path from a message intermediate event exists, and immediately after the flow object directly preceding the to_provide_data activity in the non-woven process. The

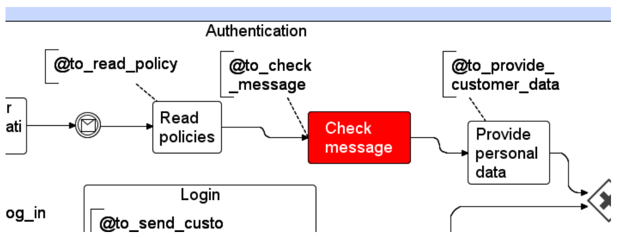
[8]For each distinct query result set $g1$, e and $sf3$ represent new instances.

sequence flow between the flow object and the to_provide_data activity is also removed. It uses the BPMN VQL PATH operator thus allowing the matching of pairs of BPMN elements connected by at least a path in the process model. The BPMN elements with white background and thinner lines, i.e., the intermediate message event, the to_provide_data activity and the flow object directly preceding the to_provide_data activity in the original process, represent the pattern to be matched against the process, i.e., the to_provide_data activities for which there exists a path from an intermediate message event and that are preceded by at least a flow object. The BPMN elements with darker background and thicker lines (i.e., the to_check_message task and its semantic annotation), represent, instead, the behaviour to be added to the process. Finally, the sequence flow crossed by the REMOVE operator is the behaviour that has to be removed. In this case, the selection sub-pattern is composed of the to_provide_data activity, the flow object directly preceding the activity and the sequence flow connecting these two elements. The select clause of the query contains, hence, the variable *?a1*, representing the to_provide_data activity, the variable *fo1*, denoting the flow object immediately preceding the to_provide_data activity in the non-woven process and *sf1* for the sequence flow. For each result of the query, the three BPM-type and the four BPM-structural assertions reported in Figure 7.10 are added to the BPKB[9]. Moreover, all the assertions involving $sf1$ are removed.

Aspects using the BPMN VQL NEST operator. Figure 7.11 shows an example of a BPMN VRL rule that, in the weaving phase, adds an end event after the to_update activities included in a to_manage sub-process. In detail, if the to_update activity (contained in a to_manage sub-process at whatever level of nesting) is followed by a data-based XOR gateway, the rule only adds an end event to the gateway alternatives; otherwise (i.e., if the activity is followed by a flow object that is not a data-based exclusive gateway), the rule removes the existing sequence flow connecting the activity with the flow object and adds a data-based XOR gateway with two outgoing sequence flows, one having as target the flow object directly following the to_update activity in the original process and, the second, connected to a new end event. It uses the BPMN VQL NEST operator thus allowing the matching of BPMN elements nested in sub-processes. The BPMN elements with white background and thinner lines, i.e., the to_manage sub-process, the to_update activity and either the directly connected data-based exclusive gateway or the directly connected flow object (as well as the connecting sequence flows) represent the matching pattern. The behaviour to be added to the process, instead, is provided by the BPMN elements with darker background and thicker lines (i.e., either the end event and the connecting sequence

[9]For each distinct query result set ($a1$, $fo1$, $sf1$), $t1$, $sf2$ and $sf3$ represent new instances.





SPARQL Queries:

```
SELECT ?a1, ?fo1, ?sf1
WHERE {
  ?a1 rdf:type bpmn:activity.
  ?a1 rdf:type bdo:to_provide_data.
  ?me1 rdf:type bpmn:message_intermediate_event.
  ?me1 (bpmn:has_sequence_flow_source_ref_inv/
    bpmn:has_sequence_flow_target_ref)* ?a1. }
  ?sf1 bpmn:has_sequence_flow_target_ref ?a1. }
  ?sf1 bpmn:has_sequence_flow_source_ref ?fo1.
  ?fo1 rdf:type bpmn:flow_object }.
  ?sf1 rdf:type bpmn:sequence_flow }.
}
```

For each result $(a1, fo1, sf1)$:

ASSERTIONS TO BE ADDED:

task$(t1)$
sequence_flow$(sf2)$
has_sequence_flow_source_ref$(sf2, fo1)$
has_sequence_flow_target_ref$(sf2, t1)$
sequence_flow$(sf3)$
has_sequence_flow_source_ref$(sf3, t1)$
has_sequence_flow_target_ref$(sf3, a1)$

INSTANCES TO BE REMOVED:

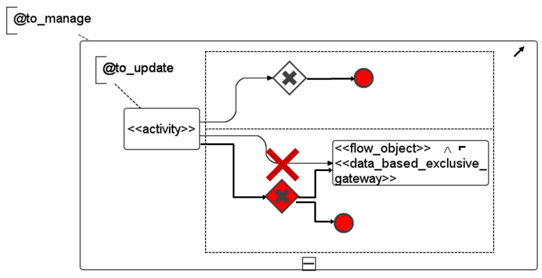
$sf1$

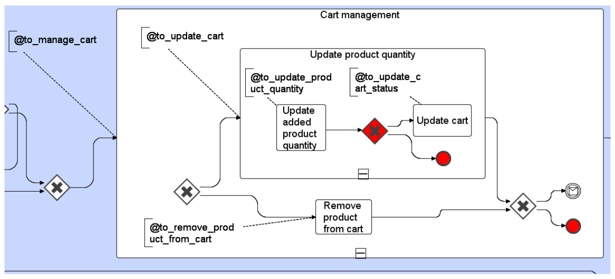
Figure 7.10: Example of a BPMN VRL rule using the BPMN VQL PATH operator: after the weaving phase, immediately before all the to_provide_data activities for which there exists at least a path from a message intermediate event, the "Check message" task is executed.

flow, in the first OR alternative, or the data-based exclusive gateway, the end event and the three sequence flows in the second OR alternative). Finally, the sequence flow crossed by the REMOVE operator represents the behaviour to be removed from the process, whenever the second of the OR alternatives matches (i.e., when the to_update activity is not directly followed by any data-based exclusive gateway). The selection sub-pattern is composed of the to_update activity, the data-based XOR gateway, the flow object directly following the to_update activity and the sequence flow connecting the to_update activity to the flow object. The select clause of the query contains, hence, the variable *?a1* representing the to_ask_for activity, the variable *?g1* representing the gateway, when the first pattern matches, and *?a2* and *?sf2* when the second OR alternative matches. For each result of the query, either the two BPM-type and the three BPM-structural assertions (in case $g1$ is valued), or the five BPM-type and the six BPM-structural assertions (in case $a2$ is valued) reported in Figure 7.11 are added to the BPKB[10]. Moreover, all the BPM-type, BPM-structural and BPM-semantic assertions involving $sf2$, if this is valued, are removed.

BPMN VRL was designed to guarantee an easy use (in terms of, for example, limited learning effort, simple aspect understanding) of the aspectization mechanism for business designers, favouring a simple semantics of the language over a very powerful expressivity. An example of a possible limitation of the BPMN VRL expressivity is its inability to distinguish among different "update" parts of a rule (e.g., behaviour addition and removal, or many behaviour removals), when the cardinality of the variables (characterizing the different "update" parts) of the selection sub-pattern is different. For instance, we can consider the aspect requiring the removal of all the outgoing sequence flows from a data-based exclusive gateway and the introduction of new flows (e.g., a new sequence flow connected to a "Read paper" task and another connected to an end event). Such an aspect can not be represented with only one BPMN VRL rule (as done in the BPMN VRL rule in Figure 7.12) unless as many pairs of tasks and end events (and related sequence flows) as the outgoing sequence flows are introduced in the woven process. In the example, the designer is supposed to specify two rules: the first removing all the sequence flows outgoing from the data-based exclusive gateway (Figure 7.13a), and the second adding two outgoing sequence flows (one connected to a "Read paper" task and the other connected to an end event) to the gateway (Figure 7.13b). We think the relatively simple semantics of BPMN VRL compensates its limited expressivity. Often, splitting the intended update part into two or more rules is sufficient to achieve the designer's purpose.

[10]For each distinct query result set ($a1$, $g1$, $a2$, $sf2$), e and $sf4$ or e, $sf4$, $sf5$ and $sf6$ represent new instances if $g1$ or $a2$ is valued, respectively.





SPARQL Queries:

```
SELECT ?a1 , ?g1, ?a2 , ?sf2
WHERE {
  ?a1 rdf:type bpmn:activity.
  ?a1 rdf:type bdo:to_update.
  { ?sp1
  bpmn:has_embedded_sub_process_sub_graphical_elements+
      ?a1.
  ?sp1 rdf:type bpmn:embedded_sub_process.
  ?a1 rdf:type bdo:to_manage.
  { ?sf1 bpmn:has_sequence_flow_source_ref ?a1.
   ?sf1 bpmn:has_sequence_flow_target_ref ?g1. }
   ?g1 rdf:type bpmn:data_based_exclusive_gateway.
   ?sf1 rdf:type bpmn:sequence_flow }.
UNION
  { ?sf2 bpmn:has_sequence_flow_source_ref ?a1.
   ?sf2 bpmn:has_sequence_flow_target_ref ?a2.
   ?sf2 rdf:type bpmn:sequence_flow }.
   ?a2 rdf:type bpmn:flow_object.
   ?sf3 bpmn:has_sequence_flow_target_ref ?a1.
   ?sf3 bpmn:has_sequence_flow_target_ref ?a3.
   ?sf3 rdf:type bpmn:sequence_flow }.
   ?a3 rdf:type bpmn:data_based_exclusive_gateway.
   FILTER (?a2!= ?a3) }
}
```

For each result ($a1$, $g1$, $a2$, $sf2$):
ASSERTIONS TO BE ADDED:
if $g1$ is valued
end_event(e)
sequence_flow($sf4$)
has_sequence_flow_source_ref($sf4$, $g1$)
has_sequence_flow_target_ref($sf4$, e)
if $a2$ is valued
data_based_exclusive_gateway(g)
end_event(e)
sequence_flow($sf4$)
has_sequence_flow_source_ref($sf4$, $a1$)
has_sequence_flow_target_ref($sf4$, g)
sequence_flow($sf5$)
has_sequence_flow_source_ref($sf5$, g)
has_sequence_flow_target_ref($sf5$, $a2$)
sequence_flow($sf6$)
has_sequence_flow_source_ref($sf6$, g)
has_sequence_flow_target_ref($sf6$, e)
INSTANCES TO BE REMOVED:
if $sf2$ is valued
$sf2$

Figure 7.11: Example of a BPMN VRL rule using the BPMN VQL NEST operator: after the weaving phase, it will be possible to end the current sub-process from all the to_update activities included in a to_manage sub-process, at whatever level of nesting.

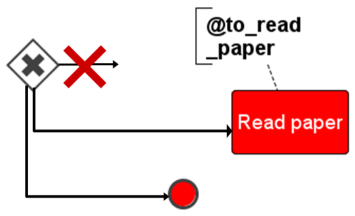


Figure 7.12: Example of a BPMN VRL rule for which the different "update" parts (i.e., the task and the end event addition and the sequence flow removal) could have a different cardinality (i.e., one for the addition and X for the removal, according to the number of outgoing sequence flows in the non-woven process). The matched gateway in the woven process will contain as many outgoing to_read_paper and end events as the sequence flows outgoing the gateway in the non-woven process.

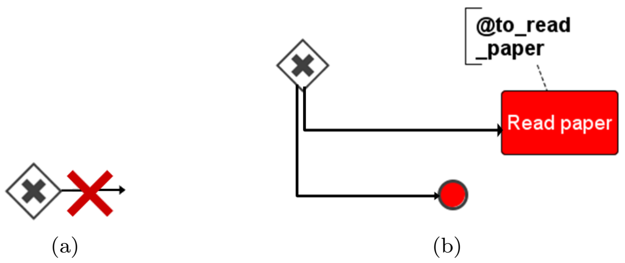


Figure 7.13: Example of two BPMN VRL rules that, if applied in sequence, replace all the outgoing sequence flows of data-based exclusive gateways with two sequence flows, one connected to a "Read paper" task and the other connected to an end event.

We developed a tool for the automated weaving of aspects in the BPKB (*aspectization tool*). It takes advantage of the ARQ[11] implementation of SPARQL1.1.[73] and of the population tool for, respectively, querying the BPKB (and hence retrieving the BPD instances that satisfy the matching criterion) and for managing the modifications in the rule advice. It uses the `org.w3c.dom` XML parsing library to manage the aspect file, the Jena[12] API for query formulation and the ARQ engine for query execution, Protege[13] libraries to populate the resulting OWL Abox, and Pellet[14] for reasoning. Once business experts have separately modelled the desired aspects as BPMN VRL rules, they can generate the woven process by providing the aspects as input to the aspectization tool, that parses them and populates the BPKB accordingly.

[11]http://jena.sourceforge.net/ARQ/
[12]http://jena.sourceforge.net/
[13]http://protege.stanford.edu/
[14]http://clarkparsia.com/pellet/

7.2 Exception Handling Aspectization

In business processes, exception handling represents a typical example of crosscutting concern. The exception handler can be tangled in different scattered points of the process, thus increasing the process complexity, when explicitly managed. Hence, though processes meeting exception handling requirements have higher robustness ([43]), business designers often focus only on the "happy path", to make the process model easier to understand.

Aspects offer the possibility to take out the complexity added to the "happy path" by the exception handling. Business experts can modularize exception handlers into aspects defined in BPMN VRL, thus separating them from the "happy path" and hence ensuring a better readability. Aspects can then be woven only when needed, e.g., for constraint verification of the whole process.

For example, as observed in Chapter 5, modelling of the on-line shopping process depicted in Figure 7.4 may need the definition of exception handling requirements to be verified on the process itself, as for example[15]:

(a) Existence of product unavailability exception:
the activity of reserving products in the On-line Shop pool has always to catch a "product unavailability" error event;

(b) Handling of product unavailability:
the "product unavailability" error event caught by the activity of reserving products in the On-line Shop pool has to be handled by executing in parallel two activities. The first one is an activity for warning the buyer; the second one is a sub-process for ordering the unavailable products;

(c) Handling of compulsory-login failure:
the activity of "sending customer data" in the "log-in" process has always to allow receiving a "compulsory-login failure" error event from the On-line Shop pool. The "log-in" process has, in turn, always to catch this error and the error event has to be handled by stopping the process;

The verification of these constraints can lead to the detection of violations (as in the case of the on-line shopping process in Figure 7.4), which demand for exception handling mechanisms. Aspects (and BPMN VRL rules) can be exploited for the modularization of the exception handlers.

[15]For completeness, we recall here the same exception handling constraints reported in Chapter 5.

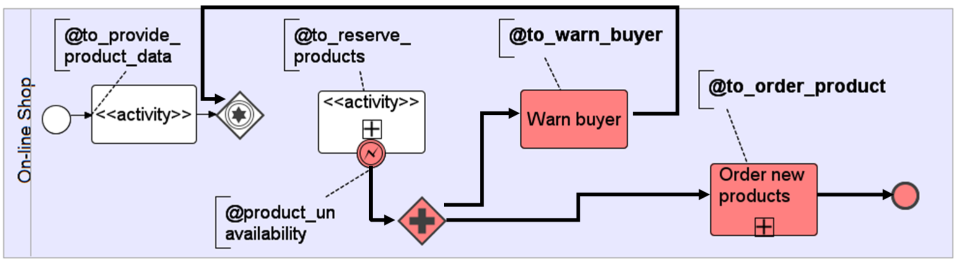


Figure 7.14: Product unavailability aspect

For example, Figure 7.14 reports a BPMN VRL aspect describing a possible exception handling (compliant with requirements (a) and (b)) for the resource unavailability exception. A product_unavailability intermediate error event has to be added on the boundary of activities reserving products in the On-line Shop pool, and it has to be handled by warning the buyer and ordering the unavailable products. The matching criterion looks for process elements that require exception handling mechanisms: it intercepts occurrences of On-line Shop activities reserving products (i.e., annotated with the semantic concept to_reserve_products). Moreover the matching criterion locates the process elements required by the aspect advice (e.g., the event-based exclusive gateway originating from the start event of the On-line Shop pool). The advice of the aspect describes the catching of the exception and its management: the caught exception is handled by means of a parallel gateway with two outgoing edges, one connected to a to_order_product sub-process (followed by an end event) and the other connected to a to_warn_buyer activity followed by the initial event-based gateway.

Figure 7.15 describes a second example of aspect related to the log-in failure exception (according to the exception handling requirement (c)). This rule has a more complex semantics than the one of the previous rule and it involves both the Customer and the On-line Shop pool.

With regards to the On-line Shop pool, the pointcut intercepts the occurrences of the activities checking customer data, contained in an authentication sub-process and for which there exists at least a path (the sequence flow with double head) from a message intermediate event (contained in the same authentication sub-process) generated by the to_send_customer_data activity in the to_log_in sub-process of the Customer pool. Once all the occurrences of activities matching this criterion have been identified, the action to take is chosen according to the type of flow object following the activity checking the customer data. Three possible different cases may occur (see the three cells of the dotted table in the On-line Shop pool in Figure 7.15):

(a) the activity checking the customer data is directly followed by a flow object that is not a data-based exclusive gateway (top cell of dotted table in Figure 7.15). In this case: (i) the sequence flow connecting the to_check_customer_data activity and the non-gateway flow object is removed (filled cross); (ii) a data-based exclusive gateway is added; (iii) the new gateway is connected through an incoming sequence flow to the to_check_customer_data activity; (iv) the new gateway is connected through two outgoing sequence flows to the non-gateway flow object and to a new activity notifying the login failure to the Customer pool, respectively.

(b) the to_check_customer_data sub-process is followed by a split data based exclusive gateway (i.e., an exclusive gateway with only an incoming edge). The middle cell of the dotted table in Figure 7.15 is matched, hence only a sequence flow connecting the gateway to the new activity notifying the login failure is added.

(c) the to_check_customer_data activity is directly followed by a data-based exclusive gateway with more than one incoming sequence flow (bottom cell of dotted table in Figure 7.15). As in the first alternative: (i) the sequence flow connecting the to_check_customer_data activity and the data-based exclusive gateway is removed (filled cross); (ii) a new split data-based exclusive gateway is added; (iii) the new gateway is connected to the to_check_customer_data activity; and (iv) the new gateway is connected to the original exclusive gateway and to the new "Notify login failure" activity, respectively.

In all three cases, the added "Notify login failure" activity is followed by a login_failure end event, terminating the sub-process in which it is contained.

In the Customer pool, instead, the aspect manages the login_failure intermediate event generated by the On-line Shop pool. Similarly to the On-line Shop pool, according to the type of flow object immediately following the to_send_customer_data activity in the to_log_in sub-process, a different behaviour is specified in order to catch the failure event. Moreover, the to_log_in sub-process containing the received login_failure event has to catch the event and manage it by terminating the (sub-)process.

At weaving time, all the scattered occurrences captured by the quantification part of each rule will be identified and modified according to the corresponding advice. For example, the aspect handling the product unavailability will exploit the process semantic information in order to match the sub-process "Reserve product/s" and it will apply on it the corresponding modifications (as shown in Figure 7.16).

In order to make aspects reusable across processes, generic exception handlers may be defined for a class of activities that are likely to require that kind of handlers. Then,

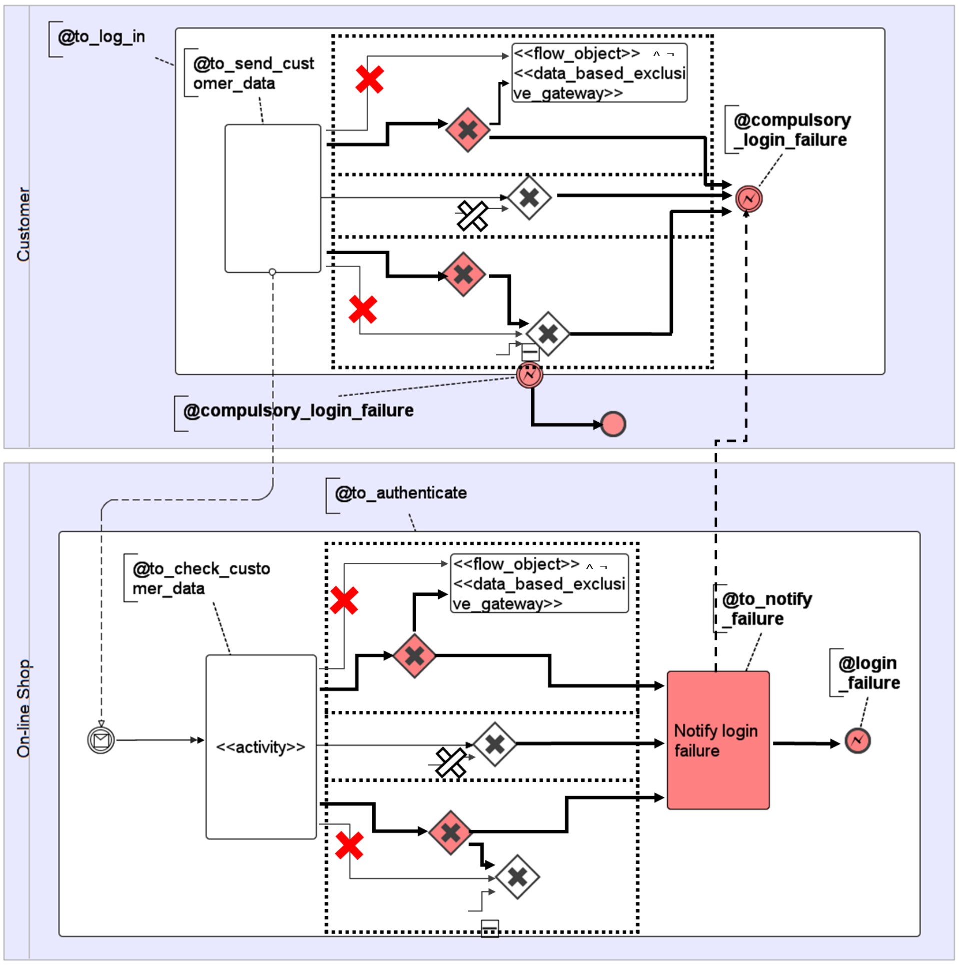


Figure 7.15: Compulsory log-in failure aspect

specific aspects are specializations of such exception handler categories, defined according to the specific needs of the designers and of the process itself. For example any to_log_in activity is likely to require a login_failure handler, that can be specialized into different aspects. Well known exception handling strategies could be specified as reusable BPMN VRL aspect libraries.

7.2.1 Using semantic constraints to support aspect definition

After defining the exception handling aspects in BPMN VRL, business designers can verify that exception handling requirements (e.g., the same requirements that could have led to the detection of exception handling constraint violations and, hence, to the aspect definition, as for example the requirement (a) for the process in Figure 7.4), translated

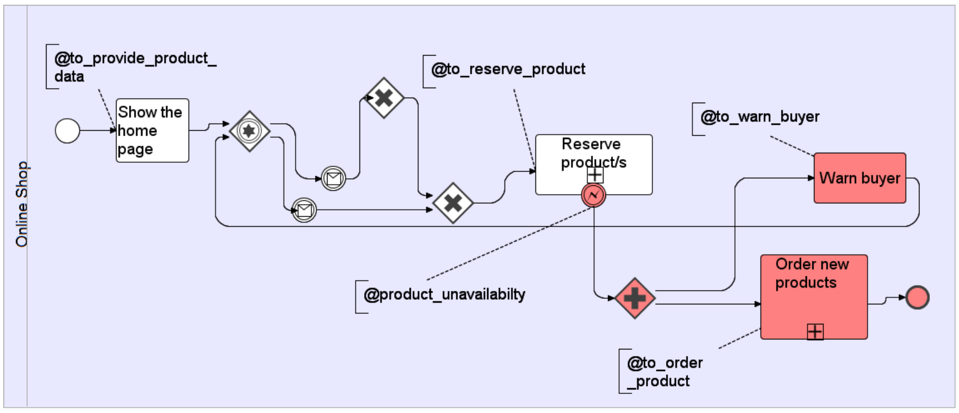


Figure 7.16: Partial view of the on-line Shop process in Figure 7.4 in which the aspect in Figure 7.14 has been woven. For space reasons only the parts of the process affected by the aspect are reported.

into semantic constraints, are satisfied on the woven process. In addition to that, the exception handling constraints, formalized before the aspect definition, can be useful to support business designers also in the modelling of the exception handling aspects. Semantic constraints are thus useful both before and after aspects are defined by business designers.

The definition of exception handling aspects can be based on the verification of the constraints originated from the exception handling requirements. In fact, constraint verification does not only return the process occurrences violating the constraints (if any), but it also suggests where modifications have to be applied in order to solve constraint violations. For instance, given a violated inclusion axiom, by exploiting the concept on the left of the inclusion as matching criterion and the part on the right for the advice, a *skeleton aspect* (i.e., a starting point for the definition of the exception handling aspect) can be automatically generated. For example, in the process in Figure 7.4, the requirement (a) is violated by the sub-process “Reserve product/s”. A skeleton aspect can be automatically generated starting from the violated inclusion axiom (7.4) (already formalized in Chapter 5):

$$
\begin{aligned}
&\mathsf{reserveProductOn\text{-}line} \equiv \\
&\quad \mathsf{BPMNO{:}activity} \sqcap \\
&\quad \mathsf{BDO{:}to\_reserve\_product} \sqcap \exists \mathsf{has\_graphicals}^{-}. \\
&\qquad \exists \mathsf{has\_process\_ref}^{-}.\mathsf{BDO{:}on\text{-}line\_shop} \\
&\mathsf{reserveProductOn\text{-}line} \sqsubseteq \exists \mathsf{BPMNO{:}has\_target}^{-}. \\
&\quad (\mathsf{BPMNO{:}error\_intermediate\_event} \sqcap \mathsf{BDO{:}product\_unavailability})
\end{aligned}
\tag{7.4}
$$

The generated aspect will capture all the reserve_product_On-line instances (i.e., all the to_reserve_product activities in the On-line Shop pool) and it will add a product_unavailability intermediate event on its boundary. By taking advantage of the visualization of the process elements violating the constraints, the business designer can complete this skeleton aspect in order to handle the exception according to the requirements. The skeleton aspect automatically produced from the violated constraint can describe the exception handling with different levels of detail according to the type of violated constraint, ranging from just exception catching ("where?") to the complete exception handling ("what?"), as in case of the skeleton aspect that is generated by the violation of the constraint (b). Constraint verification on the woven process will finally check whether all the constraint violations have actually been solved.

7.3 Performance Evaluation

In order to provide a first evaluation of the applicability for business designers of the proposed aspect-oriented approach, we performed an experiment[16] to evaluate its performance in one of its main use: the exception handling management. In the experiment, we considered six different processes of increasing size (the same used for the BPMN VQL evaluation in Chapter 6 and for the constraint verification evaluation in Chapter 5) and, for each of them, an exception handling requirement (of the same kind of requirement (a)). The purpose of this experiment was to investigate the performance of the exception handling management approach (including both the detection of violations of exception handling constraints and the modularization of exception handling requirements into aspects) proposed in Section 7.2, and, in general, the applicability of business process aspectization, as the size of the BPKB increases.

[16]The machine used for the experiment is a desktop PC with an Intel Core i7 2.80GHz processor, 6 Gb of RAM, and running Linux RedHat.

In detail, first we validated the BPKB (populated with the BPD graphical objects) against each constraint considered, to check whether the given process satisfies or not the requirement (*Validation Phase I*). Once the appropriate aspect handlers have been selected for the process concerns violating the requirements, we ran the aspectization tool to weave the aspects in the main process (*Aspectization Phas*e). Finally, we validated again the BPKB against the considered constraint, to check whether the woven process satisfied the exception handling requirement imposed on the process (*Validation Phase II*). The reasoning tasks required in each phase have been performed with the support of the Pellet reasoner (v2.0.2), integrated with the Pellet IC Validator (v0.4) for the constraint validation tasks. For each phase, the average time spent over 100 runs and the corresponding standard deviation have been computed.

		P1	**P2**	**P3**	**P4**	**P5**	**P6**
Process	*Graphical Process Elements*	92	175	237	327	387	475
Domain Ontology	*Classes*	124	124	101	114	79	124
	Class Axioms	133	133	101	113	77	133
Validation Phase I	*Constraint Validation Time(s)*	6.357 (0.312)	10.532 (0.480)	14.300 (0.969)	16.064 (1.262)	26.008 (5.922)	37.596 (4.775)
Aspectization Phase	*Added Graphical Elements*	8	16	9	16	27	22
	Added Individuals	16	32	18	40	54	48
	Added Assertions	50	95	54	117	162	146
	Aspectization Time(s)	3.634 (0.099)	3.790 (0.095)	3.842 (0.080)	3.992 (0.116)	3.997 (0.110)	4.068 (0.095)
Validation Phase II	*Constraint Validation Time(s)*	6.681 (0.536)	10.929 (0.527)	14.770 (1.073)	16.612 (1.285)	33.491 (5.473)	39.307 (6.004)

Table 7.1: Experimental Evaluation Results. The time values reported in Tables 7.1 are in the form *avg* (*sd*), where *avg* and *sd* are respectively the arithmetic mean and the standard deviation of the execution times obtained over 100 runs on the same input data.

The results are reported in Table 7.1. As shown in the Table, the most demanding phases in terms of performance are the constraint validation ones, for which the computation time increases considerably as the size of the process (and, hence, of the BPKB) grows. The time required for the aspectization phase, instead, is reasonable for all the six processes considered and, in general, less sensitive to process size variations.

Though this first analysis is a preliminary evaluation and more replications would be required with different types of aspects, processes and usage scenarios, the obtained results indicate that the proposed business process aspectization approach (in particular the detection of exception handling constraint violations and its aspectization) is compatible with an on-line usage at modelling time by designers on small/medium size processes. This result encourages us to proceed with further analyses.

Chapter 8

Experimental Results: BPMN VQL Empirical Evaluation

"As far as the laws of mathematics refer to reality,
they are not certain, and as far as they are certain,
they do not refer to reality."
Albert Einstein

As part of the evaluation of the proposed approach, we analysed in depth one of the possible uses of business process semantic annotations considered in the thesis. In detail, we investigated the effectiveness and efficiency (in terms of benefits gained and effort required) of the BPMN VQL language for retrieving information scattered across the process and for documenting it.

In Subsection 6.1.2 of Chapter 6, we provided a first evaluation of the BPMN VQL in terms of expressive power (with respect to the other visual languages for querying business processes) and performance. In this chapter we are interested in evaluating the usefulness of the BPMN VQL in terms of benefits gained versus effort required by business experts in retrieving and documenting information scattered across semantically annotated business processes. More precisely, our aim is to compare the advantages of the adoption of BPMN VQL with respect to the base approach (i.e., using natural language) for documenting and retrieving information scattered across the process. To this purpose, we conducted an experimental study with human subjects.

In the next sections we first describe the goal and the design of the experiment (Section 8.1). We then provide the experiment results (Section 8.2) and, finally, we present some possible threats to its validity (Section 8.3) and an overall discussion (Section 8.4).

8.1 Experiment Definition, Planning and Design

In the rest of this subsection we describe the study by following the methodology presented by Wohlin [194].

8.1.1 Goal of the Study and Research Questions

The *goal* of the study is to analyse two approaches (one based on natural language queries and the other on BPMN VQL queries) with the purpose of evaluating query understandability for documentation purposes and query execution performance in the context of business process maintenance operations. The quality focus is related to the accuracy[1] of the results obtained, the time spent in matching the queries against the process and the subjective perception of the effort required during query understanding and execution. The perspective considered is of both researchers and business managers, interested in investigating the benefits of the adoption of a visual language for supporting business designers and analysts in retrieving crosscutting concerns scattered across the process and documenting them. The context of the study consists of two objects (two semantically annotated processes and the ontologies used for their annotation) and a group of researchers and PhD students working at Fondazione Bruno Kessler (FBK) as subjects.

The objective of the study is: (i) investigating the understandability of BPMN VQL queries with respect to natural language (NL) queries; and (ii) evaluating the performance (in terms of results and effort required) of BPMN VQL queries with respect to the NL for retrieving information. To this purpose, we asked the involved subjects to perform two different types of assignment: the *Query Understandability* and the *Query Execution* assignment, respectively. The *Query Understandability* assignment (aimed at comparing the ease of understanding queries in BPMN VQL and NL) consists, for both the languages, in matching the queries against the process. The *Query Execution* assignment, instead, differs depending on the language. Since the purpose of the *Query Execution* assignment is to evaluate the performance required by the query execution, it consists in matching the query against the process, in case of NL, and in formulating BPMN VQL queries (to be automatically executed by a tool), in case of BPMN VQL.

We believe that the graphical notation of BPMN VQL queries, as well as their higher formality with respect to natural language, helps designers and analysts in disambiguating and clarifying queries and hence the concerns they represent. Moreover, we also expect that formulating queries in BPMN VQL, which can be automatically executed, is easier

[1]In this chapter we will use the term accuracy with an informal meaning, i.e., with no reference to any classification measure.

than matching NL queries against the process. These two expectations provide a direction for the research questions (and the hypotheses) we are interested in investigating:

RQ1 *Are BPMN VQL queries easier to understand than natural language queries?*
RQ2 *Is BPMN VQL query formulation easier to perform as compared to matching the results of natural language queries?*

RQ1 deals with the understandability of BPMN VQL queries with respect to NL queries. The hypotheses related to this question are the following:

- ($H1_0$) When performing query understanding tasks, understanding BPMN VQL queries is not easier than understanding NL queries.
- ($H1_a$) When performing query understanding tasks, understanding BPMN VQL queries is easier than understanding NL queries.

We investigated this first research question by taking into account and inspecting three different factors:

- the (objective) impact that query understanding is expected to have on the accuracy of the results obtained (we expect higher accuracy for BPMN VQL queries);
- the (objective) effort, in terms of time, required to perform query understanding tasks (for BPMN VQL queries it would be desirable to observe a time not significantly higher than for NL queries);
- the perceived (subjective) effort required to perform query understanding tasks (we expect a lower effort for BPMN VQL queries).

Hence, $H1_a$ can be decomposed in the following three sub-hypotheses:

- ($H1_{a_A}$) The results obtained by performing BPMN VQL query understanding tasks are more accurate than those obtained performing NL query understanding tasks;
- ($H1_{a_B}$) There is no difference between the time required to perform BPMN VQL and NL query understanding tasks;
- ($H1_{a_C}$) The effort perceived when performing BPMN VQL query understanding tasks is lower than the one perceived when performing NL query understanding tasks.

RQ2 deals with the formulation of BPMN VQL queries, that can be automatically matched against the process (by means of a tool), compared to the manual matching of NL queries. Similarly to **RQ1**, the hypotheses for **RQ2** are the following:

- ($H2_0$) When performing query execution tasks, formulating BPMN VQL queries is not easier than matching NL queries against the process;
- ($H2_a$) When performing query execution tasks, formulating BPMN VQL queries is easier than matching NL queries against the process.

Also in this case, in order to deal with the research question, we considered and evaluated three main factors:

- the (objective) impact that query formulation/matching has on the accuracy of the results obtained by respectively formulating/matching the query in query execution tasks (we expect higher accuracy for BPMN VQL queries);
- the (objective) effort, in terms of time, required to perform query execution tasks (for BPMN VQL queries we expect a time not higher than NL queries);
- the perceived (subjective) effort required to perform query execution tasks (we expect a lower effort for BPMN VQL queries).

Hence, the corresponding hypotheses in which $H2_a$ can be decomposed are the following:

- ($H2_{a_A}$) The results obtained when executing the BPMN VQL queries formulated in query execution tasks are more accurate than those obtained by matching the NL queries (in query execution tasks);
- ($H2_{a_B}$) There is no difference between the time required to formulate BPMN VQL queries and matching NL queries (in query execution tasks);
- ($H2_{a_C}$) The effort perceived when formulating BPMN VQL queries is lower than the one perceived when matching NL queries against the process (in query execution tasks).

8.1.2 Context

The objects of the study are two semantically annotated business processes describing real-life procedures: *Bank Account Process* and *Mortgage Process*. The *Bank Account Process*[2] represents the exchange of information between the customer and the bank for opening and activating a bank account. It is made of 2 pools (the "Bank" and the "Customer") and it contains 30 activities, 16 events and 16 gateways. The associated

[2]The *Bank Account Process* is based on a process used as example in the book by Havey [77] (it is reported in the Havey's article "Modeling Orchestration and Choreography in Service Oriented Architecture" available at `http://www.packtpub.com/article/modeling-orchestration-and-choreography-in-service-oriented-architecture`).

	L1		*L2*	
	NL	**BPMN VQL**	**NL**	**BPMN VQL**
G_A	*Bank Account Process*			*Mortgage Process*
G_B		*Bank Account Process*	*Mortgage Process*	
G_C	*Mortgage Process*			*Bank Account Process*
G_D		*Mortgage Process*	*Bank Account Process*	

Table 8.1: Study balanced design

ontology used for its annotation contains 77 concepts and 30 of them are used for the process semantic annotation. The *Mortgage Process*[3] is instead a process describing the procedure regulating the acceptance or the refusal by the "Mortgage Co." company of mortgage requests formulated by potential customers. It is also made of two pools (the "Mortgage Co." and the "Potential Customer") and it is slightly larger than the *Bank Account Process*: it contains 35 activities, 26 events and 18 gateways. The associated ontology has 99 concepts and 31 of them are used for semantically annotating the process.

The subjects involved in the study were 12 persons working at FBK in the domain of software engineering or knowledge management: 5 PhD students and 7 researchers.

8.1.3 Design, Material and Procedure

The design adopted in this study is a *balanced design* [194]. Subjects are divided into four groups (*G_A*, *G_B*, *G_C* and *G_D*) and asked to perform two types of assignment (*Query Understandability* and *Query Execution*) on two different objects (*Bank Account Process* and *Mortgage Process*) with two *treatments* (NL or BPMN VQL queries) in two laboratory sessions (*L*1 and *L*2). Each group worked with both treatments and with both objects, by performing both the *Query Understandability* and *Query Execution* assignment on one process with the NL queries in one laboratory and on the other process with BPMN VQL queries in the other laboratory. In detail, the schema adopted in the study is reported in Table 8.1. Such a schema allows to limit the impact of the learning effect on the objects and to limit possible undesired effects on the results due to the learning effect on the treatment.

During the experiment, subjects received the following material[4] to perform the required tasks:

- a pre-questionnaire collecting information about knowledge and experience of subjects (reported in Appendix A);

[3]The *Mortgage Process* is based on a process used as running example in the BPMN book by White et al. [192]

[4]The experimental package (containing the material used in the experiment) is available on-line at `http://selab.fbk.eu/difrancescomarino/BPMNVQLEval` for repetition purposes.

- a BPMN quick handbook for recalling the main constructs of the language to novel users;
- a BPMN VQL handbook for recalling the main notions of the query language (when needed);
- a semantically annotated process (the *Bank Account Process* or the *Mortgage Process*);
- the ontology used for annotating the process;
- an extract of the BPMN ontology for clarifying relationships among BPMN constructs;
- a description of the tasks to perform, i.e., either:
  - 10 queries: 6 NL queries to be matched against the process for the *Query Understandability* assignment and 4 NL queries to be matched against the process for the *Query Execution* assignment; or
  - 10 queries: 6 BPMN VQL queries to be matched against the process for the *Query Understandability* assignment and 4 queries described in natural language to be translated into BPMN VQL queries for the *Query Execution* assignment;
- the *answer book* for reporting the answers related to the 10 required tasks. It is a set of 10 sheets, each reproducing the same annotated process and devoted to report answers for the corresponding task;
- a post-questionnaire investigating personal judgements about the executed tasks, as well as the general impression deriving from the used approach (reported in Appendix B);
- a final post-questionnaire investigating subjective judgement about the BPMN VQL benefits versus effort (reported in Appendix B).

Before the experiment execution, subjects were trained on BPMN, ontologies, semantic annotation of business processes and BPMN VQL. Moreover, subjects were also provided with a first description of the object processes in order to be able to easily understand the domain.

After the training session, in the first laboratory, subjects were asked to fill a pre-questionnaire. Then, for each laboratory, the procedure described in Figure 8.1 was followed. For both the assignments, subjects were asked to mark the starting time before executing each task and the ending time after the task execution. The *Query Understandability* consists, for both the treatments, in matching the queries against the process. The *Query Execution* for NL queries, instead, differs from the BPMN VQL *Query Execution*. The reason is that our goal is to evaluate, for each of the two approaches, the effort required for executing the query. Hence, in case of NL queries, we have to evaluate the effort required in matching the NL query, while, in case of BPMN VQL queries, we have to evaluate the effort required in formulating the BPMN VQL query, by assuming that the query is then executed by an automatic tool at no cost. Moreover, the automatic execution of the query, by allowing to visualize the retrieved results and hence to potentially reveal possible false positives and true negatives captured by the formulated query, would further support users in query refinement.

The use of the tool in the experiment, however, would have been not completely fair with respect to the manual natural language matching. Hence, we decided to partially penalize subjects involved in the BPMN VQL treatment. We allowed them to match the formulated BPMN VQL query, in order to verify its correctness, only manually, and we asked them to refine it, if necessary, by marking an estimation of the percentage of time spent in matching the query.

Finally, at the end of both laboratory sessions, subjects were asked to fill a final post-questionnaire.

8.1.4 Variables

The independent variable considered in the study is the type of query language used for performing the assignments. The independent variable, hence, can assume only two values, i.e., the two treatments: NL or BPMN VQL.

The number of dependent variables in the study, instead, is higher since for the evaluation of the two research questions we analysed both objective and subjective factors. In detail, we used the accuracy of the results of the *Query Understandability* and *Query Execution* assignments as well as the time spent to perform the tasks as objective measures. The personal judgements expressed by subjects about the effort required by tasks as subjective measures.

The set of dependent variables defined to answer the two research questions, as well as the corresponding descriptions, are reported in Table 8.2. For each of the two hypotheses, $H1$ and $H2$ (answering the research questions **RQ1** and **RQ2**, respectively), the related sub-hypotheses described in Subsection 8.1.1 have been considered (column “Sub-hp”

```
1. Query understanding assignment.
   For each task (i.e., for each NL or BPMN VQL query matching):
   1.1. mark the starting time;
   1.2. read the (NL or BPMN VQL) query;
   1.3. match the query;
   1.4. mark the ending time.
2. Query execution assignment.
   2.a. In case of NL queries, for each task
       (i.e., for each NL query matching):
           2.a.1. read the query;
           2.a.2. mark the starting time;
           2.a.3. match the query;
           2.a.4. mark the ending time.
   2.b. In case of BPMN VQL queries, for each task
        (i.e., for each BPMN VQL query formulation)
           2.b.1. read the query;
           2.b.2. mark the starting time;
           2.b.3. formulate the initial BPMN VQL query;
           2.b.4. match the formulated query;
           2.b.5. mark the initial query ending time;
           2.b.6. refine the BPMN VQL query;
           2.b.7. mark the ending time.
  3. fill the post-questionnaire.
```

Figure 8.1: Detailed study procedure (followed in each of the two laboratory sessions)

in Table 8.2). In turn, each sub-hypothesis has been further decomposed, according to the different measures (e.g., precision, recall) considered for its evaluation, into detailed sub-hypotheses (column "Det. sub-hp" in Table 8.2), each corresponding to a dependent variable (column "Variable" in Table 8.2).

In detail, we evaluated the results obtained in the *Query Understandability* assignment in order to investigate the query language understandability **RQ1** and those obtained in the *Query Execution* assignment for **RQ2**. For the evaluation of the accuracy of the task results, we exploited two metrics widely used in Information Retrieval: precision and recall. In case of the *Query Understandability* assignment, for each subject s_j and for each query q_i, we identified the set of correct results (CR_{q_i}) of the query q_i and the set of results reported by the subject s_j for the query q_i (RR_{q_i,s_j}). In case of the *Query Execution* assignment, instead, CR_{q_i} and RR_{q_i,s_j} are identified for each NL query q_i and subject s_j exactly as above, while the corresponding values, for each task t_i with the BPMN VQL treatment, are collected by automatically executing the corresponding BPMN VQL query q_i formulated by the subject s_j.

Starting from these values we computed precision P_{q_i,s_j} and recall R_{q_i,s_j} for each query

Hp	Sub-hp	Det. sub-hp	Variable	Unit/Scale	Description
$H1$	$H1_A$	$H1_{A_p}$	P_{QU}	$[0,1]$	precision
		$H1_{A_r}$	R_{QU}	$[0,1]$	recall
		$H1_{A_{fm}}$	FM_{QU}	$[0,1]$	f-measure
	$H1_B$	$H1_{B_t}$	T_{QU}	sec.	time
	$H1_C$	$H1_{C_{pequ}}$	$PEQU$	$[0,4]$	perceived effort in query understanding
		$H1_{C_{peou}}$	$PEOU$	$[0,4]$	perceived effort in ontology understanding
		$H1_{C_{peqm}}$	$PEQM$	$[0,4]$	perceived effort in query matching
$H2$	$H2_A$	$H2_{A_p}$	P_{QE}	$[0,1]$	precision
		$H2_{A_r}$	R_{QE}	$[0,1]$	recall
		$H2_{A_{fm}}$	FM_{QE}	$[0,1]$	f-measure
	$H2_B$	$H2_{B_t}$	T_{QE}	sec.	time
	$H2_C$	$H2_{C_{peqe}}$	$PEQE$	$[0,4]$	perceived effort in query execution
		$H2_{C_{pesu}}$	$PESU$	$[0,4]$	perceived effort in specification understanding

Table 8.2: Dependent variable description

q_i and for each subject s_j as follows:

$$P_{q_i,s_j} = \frac{\left|CR_{q_i} \cap RR_{q_i,s_j}\right|}{\left|RR_{q_i,s_j}\right|} \tag{8.1}$$

and

$$Rq_i, s_j = \frac{\left|CR_{q_i} \cap RR_{q_i,s_j}\right|}{\left|CR_{q_i}\right|} \tag{8.2}$$

Finally, in order to obtain a comprehensive measure for the evaluation of the assignments, we computed the F-Measure FM_{s_j,q_i}, i.e., the harmonic mean of (8.1) and (8.2):

$$FM_{q_i,s_j} = \frac{2 * P_{q_i,s_j} * Rq_i, s_j}{(P_{q_i,s_j} + Rq_i, s_j)}. \tag{8.3}$$

The formula for the F-Measure can also be rewritten by considering for each query q_i and for each subject s_j the set of correct reported results ($CRR_{q_i,s_j} = CR_{q_i} \cap RR_{q_i,s_j}$), the set of the incorrect reported results ($IRR_{q_i,s_j} = RR_{q_i,s_j} \setminus CR_{q_i}$) and the set of the unreported correct results ($UCR_{q_i,s_j} = CR_{q_i} \setminus RR_{q_i,s_j}$). Given these sets, the set of the retrieved results can be seen as the union of the set of correct reported results and the set of the incorrect reported results, i.e., $RR_{s_j,q_i} = CRR_{q_i,s_j} \cup IRR_{q_i,s_j}$ (and, hence, since the two sets are disjointed $\left|RR_{s_j,q_i}\right| = \left|CRR_{q_i,s_j}\right| + \left|IRR_{q_i,s_j}\right|$). Similarly, the set of correct results for the query q_i (CR_{q_i}) can be seen as the union of the set of correct results reported by a subject s_j and the set of the unreported correct results by the same subject, i.e., $CR_{q_i} = CRR_{q_i,s_j} \cup UCR_{q_i,s_j}$ (and, hence, since the two sets are disjointed $\left|CR_{q_i}\right| = \left|CRR_{q_i,s_j}\right| + \left|UCR_{q_i,s_j}\right|$). The resulting F-Measure formula will hence be:

$$FM_{q_i,s_j} = \frac{2 * P_{q_i,s_j} * Rq_i, s_j}{(P_{q_i,s_j} + R_{q_i,s_j})} \tag{8.4}$$

$$= \frac{2 * \left|CRR_{q_i,s_j}\right|}{2 * \left|CRR_{q_i,s_j}\right| + \left|IRR_{q_i,s_j}\right| + \left|UCR_{q_i,s_j}\right|} \tag{8.5}$$

This formula allows to have a defined F-Measure even in cases of undefined values of precision or recall. In fact, in our study, in few cases in which subjects found no results for a query (i.e., $RR_{q_i,s_j} = \emptyset$), we had undefined values for precision. In the literature these situations are faced in different ways. When possible, the value of the precision is left undefined (NaN). In other cases, in which the value is required for further computations, as in our study, either the undefined value is discarded from the computation or it is evaluated to 1 [151]. Finally, there are works in which, P_{s_j}, R_{s_j} and FM_{s_j} are computed starting from $\left|CRR_{s_j}\right| = \sum_i \left|CRR_{q_i,s_j}\right|$, $\left|RR_{s_j}\right| = \sum_i \left|RR_{q_i,s_j}\right|$ and $|CR| = \sum_i |CR_{q_i}|$ (e.g., [8, 13]). In our study we chose to compute precision by taking out the undefined values. However, we also carried out the analyses considering the few cases of undefined values for precision as 1 and we found almost no difference with the analyses performed by discarding the same values. Moreover, since in our study the size of the set of correct results is not the same for all the queries and each query is read and interpreted independently by subjects, we discarded also the last solution from the literature and we chose to evaluate each query separately, by computing precision and recall for each of them. This approach avoids to penalize and emphasize too much possible misunderstandings/perfect understandings in the specification of the queries with a high number of correct results.

In order to get a global result for each assignment k (i.e., $k \in \{$*Query Understandability*, *Query Execution*$\}$), and for each subject s_j, we computed (and we draw the boxplots of) the average of the three values (P_{k,s_j}, R_{k,s_j} and FM_{k,s_j}) over all the queries in the set of queries of the assignment k (Q_k), i.e.:

$$P_{k,s_j} = \frac{\sum_{q_i \in Q_k} P_{q_i,s_j}}{|Q_k|} \tag{8.6}$$

$$R_{k,s_j} = \frac{\sum_{q_i \in Q_k} Rq_i, s_j}{|Q_k|} \tag{8.7}$$

and

$$FM_{k,s_j} = \frac{\sum_{q_i \in Q_k} FMq_i, s_j}{|Q_k|} \tag{8.8}$$

In the objective evaluation, the time spent for completing the assignment is also considered. In case of **RQ1**, the time required for performing the *Query Understandability* assignment is collected. In detail, for each query q_i the time spent by the subject s_j (T_{q_i,s_j}) is computed by considering both the time spent for reading the NL/BPMN VQL query and for retrieving the query results against the process. For **RQ2**, it is the time required for realizing the *Query Execution* assignment to be investigated. In case of the NL treatment, T_{q_i,s_j} represents the time required by the subject s_j for retrieving the results of the query q_i. In case of BPMN VQL queries, instead, T_{q_i,s_j} is the time spent by the subject s_j for formulating and refining the query (in case of problems emerging from the manual execution of the formulated query). The time spent in matching the query is however excluded from T_{q_i,s_j} because in principle it can be automatically performed.

As for Information Retrieval metrics, also in this case, an average value per assignment type $k \in \{$*Query Understandability*, *Query Execution*$\}$ and per subject s_j has been computed and corresponding boxplots drawn:

$$T_{k,s_j} \quad = \quad \frac{\sum_{i \in Q_k} T_{s_j,q_i}}{|Q_k|} \tag{8.9}$$

With respect to the subjective evaluation, a set of answers is collected through the post-questionnaire. In detail, each subject was asked to express her evaluation on a 5-point Likert scale (from 0 to 4, where 0 is very low and 4 is very high) about the perceived effort in query understanding ($PEQU_{s_j}$), ontology understanding ($PEOU_{s_j}$), query execution ($PEQE_{s_j}$) and specification understanding ($PESU_{s_j}$).

8.2 Experimental Results

In this section we will describe the statistical analyses performed on the collected data. For each of the variables reported in Table 8.2, we analysed:

- the influence of the main factor, i.e., the treatment;
- possible cofactors influencing the obtained results.

With respect to the main factor, due to the violation of the preconditions of parametric tests (small number of data points and non-normal distribution), we decided to apply a non-parametric test to compare the distributions of data obtained with the two different treatments. Moreover, since each of the subjects performed the assignments both with the NL and the BPMN VQL treatment, we performed a paired statistical test. Starting from these considerations, we resorted to the Wilcoxon test, a non-parametric paired test.

Finally, according to the direction of the hypotheses to verify, we opted for a one-tailed or two-tailed analysis.

In order to evaluate the magnitude of the statistical significance obtained, we computed also the *effect size*, that provides a measure of the strength of the relationship between two variables. To this purpose we used the *Cohen's d* formula (the effect size is considered to be small for $0.2 \leq d < 0.5$, medium for $0.5 \leq d < 0.8$ and large for $d \geq 0.8$):

$$d = \frac{\mu_{\textsf{NL}} - \mu_{\textsf{BPMN VQL}}}{\sqrt{(\sigma_1^2 + \sigma_2^2)/2}} \tag{8.10}$$

Such a formula is slightly modified in case of paired analyses. In this case the pooled standard deviation is replaced by the standard deviation of the difference of the two distributions σ_D, i.e.,

$$d = \frac{\mu_{\textsf{NL}} - \mu_{\textsf{BPMN VQL}}}{\sigma_D} \tag{8.11}$$

For the analysis of the cofactors, we used ANOVA. Though it is a parametric test, it is quite robust (as also confirmed by the results we obtained). In detail we used two-way ANOVA for investigating the impact of the cofactors and of their interaction with the main factor on the dependent variables.

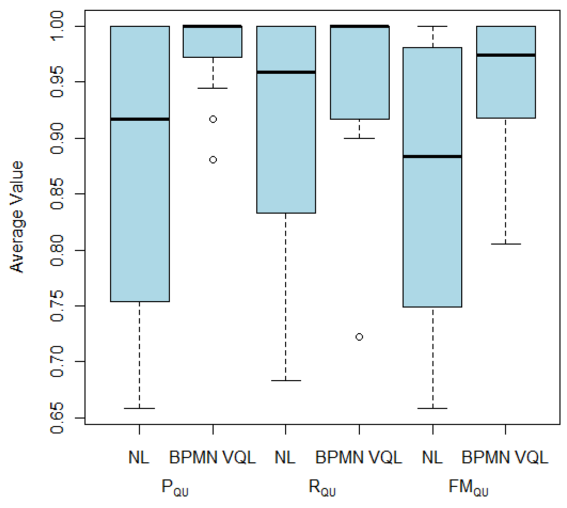
All the analyses are performed with a level of confidence of 95% (p-value < 0.05), i.e., there is only a 5% of probability that the results are obtained by chance.

8.2.1 Data Analysis

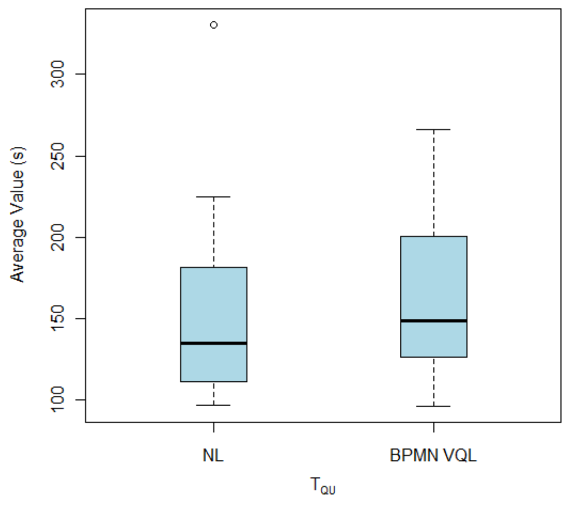
Research Question 1

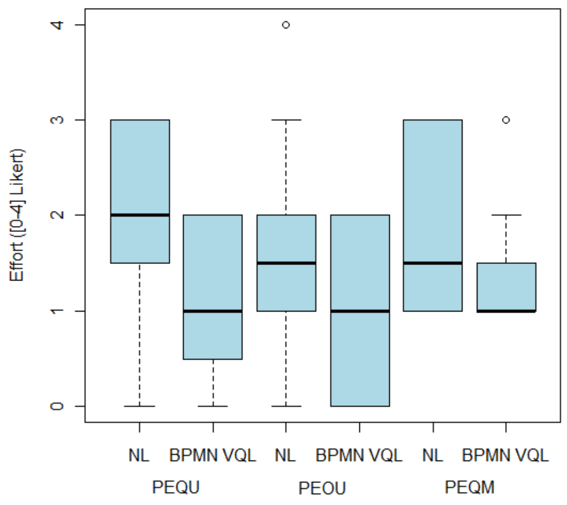
Table 8.3 reports the descriptive statistics of the data related to the **RQ1**, i.e., to the *Query Understandability* assignment of the study and Figure 8.2a reports the boxplots of precision, recall and F-Measure for the same assignment. We can notice, that, the values of precision, recall and F-Measure of the results obtained for the *Query Understandability* assignment in case of BPMN VQL queries are higher than those obtained in case of the NL queries. However, while in case of precision and F-Measure, the first quartile for BPMN VQL queries is very far from the first quartile of NL queries, the two values are much more closer in case of recall.

We applied a one-tailed Wilcoxon test in order to decide whether to reject the sub-hypothesis $H1_A$. As shown in Table 8.4, two out of the three detailed sub-hypotheses related to $H1_A$ (i.e., $H1_{A_P}$, $H1_{A_R}$ and $H1_{A_{FM}}$ in Table 8.2) could be rejected. In detail, though we were not able to reject the null hypothesis $H1_{A_R}$ (corresponding to the variable



(a) Precision, recall and F-Measure





(b) Time

(c) Query and ontology understanding as well as query matching perceived effort

Figure 8.2: Boxplots related to the *Query Understandability* assignment

Variable	**Mean**		**Median**	
	NL	BPMN VQL	NL	BPMN VQL
P_{QU}	0.870349326	0.978505291	0.916666667	1
R_{QU}	0.92037	0.951852	0.958333	1
FM_{QU}	0.861113	0.951918	0.883333	0.974074
T_{QU}	0.02.36	0.02.42	0.02.15	0.02.29
$PEQU$	2.083333333	1.08333333	2	1
$PEOU$	1.583333333	1	1.5	1
$PEQM$	1.833333333	1.33333333	1.5	1

Table 8.3: Descriptive statistics for the *Query Understandability* assignment

Variable	**Wilcoxon p-value**	**Cohen d**
P_{QU}	**0.02959**	0.698703
R_{QU}	0.1308	
FM_{QU}	**0.04118**	0.698885
T_{QU}	0.8501	
$PEQU$	**0.009864**	**0.8864053**
$PEOU$	**0.02386**	0.6479058
$PEQM$	**0.04734**	0.6267832

Table 8.4: Summary table of the results obtained by performing a paired analysis on the data related to the *Query Understandability* assignment

R_{QU}, as reported in Table 8.2), we rejected $H1_{A_P}$ (p-value = 0.02959), as well as $H1_{A_{FM}}$ (p-value = 0.02959), both with a medium effect-size value. Overall, by considering the F-Measure as a global measure of the query answers, we can reject $H1_A$, i.e, we can affirm that the use of the BPMN VQL allows to get more accurate results than the NL.

Figure 8.2b depicts the boxplot of the time spent for the *Query Understandability* assignment. In this case, for the BPMN VQL queries, the values of the time spent in understanding BPMN VQL queries are only slightly higher than those related to the time spent for understanding NL queries. We applied a two-tailed Wilcoxon test for investigating the hypothesis $H1_B$ related to the time. In this case, we were not able to reject the two-tailed null hypothesis (p-value = 0.8501), hence we cannot affirm that there is a difference of effort, in terms of time, to perform the *Query Understandability* assignment in case of NL queries and in case of BPMN VQL queries.

Finally, the boxplots of the perceived effort required for understanding queries, understanding the ontology used for the process annotation and matching the queries against the process, both for NL and BPMN VQL, are reported in Figure 8.2c. In all the three subjective ratings of the perceived effort with BPMN VQL, the boxplot is mainly concentrated in the bottom part of the plot (i.e., in the interval $[0, 2]$), meaning that subjects perceived a low/medium effort. The corresponding NL values, instead, are higher. We applied a

Variable	**Mean**		**Median**	
	NL	BPMN VQL	NL	BPMN VQL
P_{QE}	0.90625	0.989583	0.9375	1
R_{QE}	0.90625	1	0.958333	1
FM_{QE}	0.869742	0.994048	0.895833	1
T_{QE}	0.02.03	0.02.18	0.01.53	0.01.48
$PEQE$	1.833333333	1.33333333	1.5	1
$PESU$	2.083333333	1.58333333	2	1.5

Table 8.5: Descriptive statistics of the *Query Execution* assignment

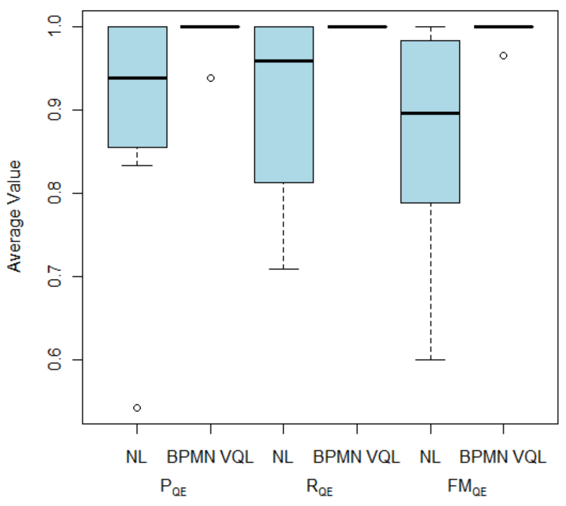
one-tailed Wilcoxon paired test in order to investigate the sub-hypothesis ($H1_C$) for all the three dependent variables. The results (p-values 0.009864, 0.02386 and 0.04734 for query understanding, ontology understanding and query matching, respectively), allows to reject each of the null hypotheses and hence the whole $H1_C$, i.e., when performing query understanding tasks, the effort perceived in understanding BPMN VQL queries is lower than the one perceived in understanding NL queries. In the specific case of query understanding, the result related to the perceived effort ($PEQU$) is also strengthened by a large Cohen d effect size ($d = 0.8864053$).

Research Question 2

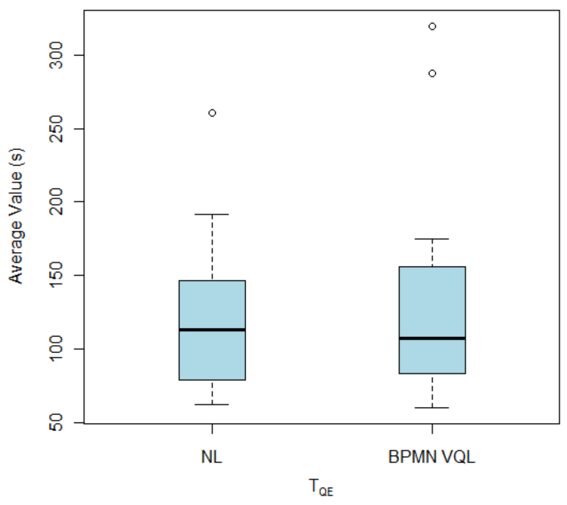
The descriptive statistics of the data related to the query execution assignment are reported in Table 8.5, while the corresponding boxplots are reported in Figure 8.3.

In detail, Figure 8.3a shows that the boxplots related to precision, recall and F-Measure obtained by automatically executing the BPMN VQL queries formulated by subjects are squeezed up to 1. On the contrary, the NL boxplots are spread across the interval $[0.7, 1]$. The BPMN VQL values for precision, recall and F-Measure are hence higher than the values obtained by the manual match of NL queries performed by subjects. The intuition suggested by the boxplots is confirmed by the one-tailed Wilcoxon test, whose results are reported in Table 8.6. In all the three cases, we are able to reject, with a level of confidence of 95%, the null hypothesis related to the specific dependent variable (p-values 0.02099, 0.01802 and 0.007001 for precision, recall and F-Measure, respectively). Moreover, in case of recall and F-Measure the result is strenghtened by the high value of the Cohen d measure. The $H2_A$ sub-hypothesis can hence be finally rejected, i.e., the results obtained by automatically executing BPMN VQL queries manually formulated by subjects are better than those obtained by manually matching NL queries against the process.

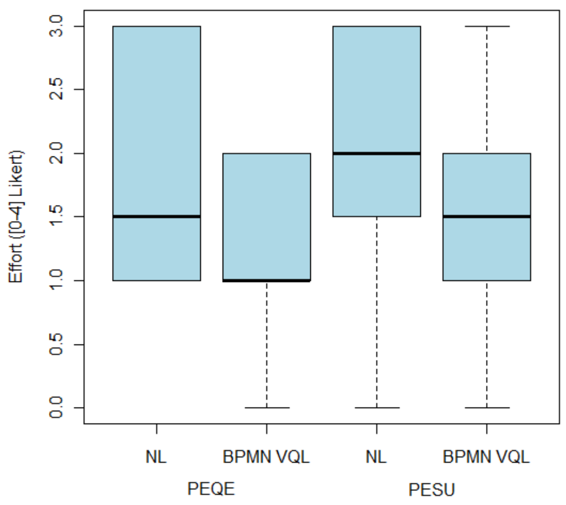
By looking at Figure 8.3b, we can observe that the time spent for formulating and



(a) Precision, recall and F-Measure



(b) Time



(c) Query execution and task specification understanding perceived effort

Figure 8.3: Boxplots related to the *Query Execution* assignment

Variable	Wilcoxon p-value	Cohen d
P_{QE}	**0.02099**	0.588606
R_{QE}	**0.01802**	**0.8436**
FM_{QE}	**0.007001**	**0.903492**
T_{QE}	0.6377	
$PEQE$	0.07023	
$PESU$	**0.02054**	0.7416198

Table 8.6: Summary table of the results obtained by performing a paired analysis on the data related to the *Query Execution* assignment

refining the BPMN VQL queries is almost the same as the time spent for matching the NL queries against the process. In Table 8.6 we can also find statistical evidence of this observation. In fact, we are not able to reject the two-tailed null sub-hypothesis $H2_B$, i.e., we cannot state that there is a difference in the time spent in matching NL queries (against the process) and the time spent for translating them in BPMN VQL.

Finally, with respect to the effort perceived when executing queries (i.e., either matching the NL queries or formulating BPMN VQL queries), in Figure 8.3c we can observe that, differently from the NL boxplot, in case of BPMN VQL queries, the boxplot is mainly distributed around low values of the Likert scale. The shape of the boxplots related to the perceived effort in understanding the natural language specification of the queries to be used for matching it against the process or for formulating the corresponding BPMN VQL query, is also similar. By applying a one-tailed Wilcoxon test, we are not able to reject the null hypothesis related to the perceived effort in query execution ($p - value = 0.07023$), but we can reject the one related to the perceived effort in specification understanding ($p - value = 0.02054$)

8.2.2 Cofactors

For both research questions and both the objective and the subjective analyses we investigated the possible impact of cofactors that, together with the main factor could have been responsible for influencing the final result. Table 8.7 summarizes the investigated factors. The cofactors analysed are mainly three:

1 the laboratory session in which the assignments have been performed;

2 the object on which the assignments have been realized; and

3 the declared experience of subjects in process modelling, with ontologies, with visual query languages and their general experience (i.e., their position).

Variable	**unit**	**description**
L	$\{L1, L2\}$	Laboratory
O	{ *Bank Account Process*, *Mortgage Process* }	Object used in the experiment
PME	$[0, 4]$	Experience in process modelling
OE	$[0, 4]$	Experience with ontologies
$VQLE$	$[0, 4]$	Experience with visual query languages
E	{ PhD, researcher }	Experience

Table 8.7: Summary table of the cofactors considered in the experiment evaluation

Laboratory (L)			
Variable	**TR**	**L**	**TR:L**
P_{QU}	**0.003802**	**0.007688**	0.112106
FM_{QU}	**0.03064**	0.27106	0.13181
T	0.78364	**0.02594**	0.54037
$PEQU$	**0.01514**	0.38635	0.66266
$PEQM$	0.11426	0.28399	**0.03957**

Process Modelling Experience (PME)			
Variable	**TR**	**PME**	**TR:PME**
P_{QU}	**0.009031**	0.303607	0.076417
FM_{QU}	**0.03677**	0.36127	0.30435
$PEQU$	**0.01514**	0.35877	0.75744

Object (O)			
Variable	**TR**	**O**	**TR:O**
P_{QU}	**0.01384**	0.38678	0.43181
FM_{QU}	**0.04369**	0.66664	0.69323
$PEQU$	**0.01690**	0.66867	1.00000

Ontology Experience (OE)			
Variable	**TR**	**OE**	**TR:OE**
P_{QU}	**0.01583**	0.93453	0.50906
FM_{QU}	**0.04193**	0.45208	0.70444
$PEQU$	**0.01514**	0.35877	0.75744

Experience (E)			
Variable	**TR**	**E**	**TR:E**
P_{QU}	**0.01619**	0.59753	0.91322
FM_{QU}	**0.03962**	0.27540	0.90538
$PEQU$	**0.01477**	0.34017	0.65685

Visual Language Experience (VLE)			
Variable	**TR**	**VLE**	**TR:VLE**
P_{QU}	**0.01569**	0.62128	0.60640
FM_{QU}	**0.0449**	0.8501	0.8116
$PEQU$	**0.01649**	0.56479	0.88966

Table 8.8: Cofactor analysis related to **RQ1**

For each of the considered cofactors we applied the two-way ANOVA in order to investigate its influence on the dependent variables, as well as the impact of its interaction with the main factor. Table 8.8 reports a selected subset[5] of the results obtained by applying the two-way ANOVA to the variables considered in **RQ1**.

We can observe that, in most of the cases, the cofactors, as well as their interaction with the treatment, has no influence on the dependent variables, thus further strengthening our results. However, there are some exceptions. For example, the time spent for performing the assignments is influenced by the laboratory session. In detail, the time spent in the second laboratory is, on average, lower than the time used in the first one, as shown in Figure 8.4.

Moreover, the interaction between the laboratory and the treatment has an impact on the perceived effort in query matching. In this case, people involved in the second

[5]We only reported results for which at least one among the main factor, the considered cofactor and their interaction has an influence on the dependent variable.

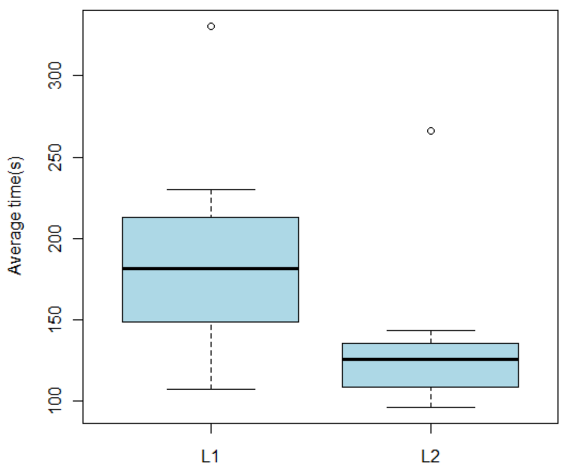


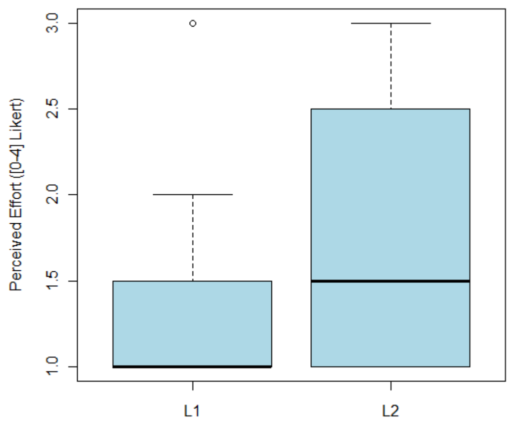
Figure 8.4: Boxplots related to the time spent in matching queries in the first and the second laboratory

laboratory perceived more effort in matching queries than subjects involved in the first one, as shown in the boxplot in Figure 8.5a. In detail, as clarified in the interaction plot in Figure 8.5b, in case of NL, the perceived effort in matching queries increases from the first to the second laboratory, while in case of BPMN VQL, it decreases.

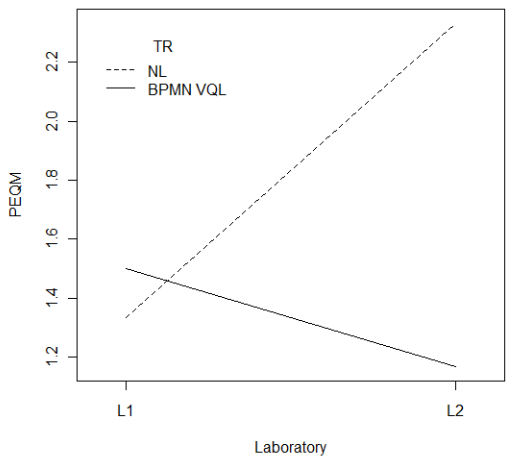
We performed the same analyses also for the variables related to the research question **RQ2**. As for **RQ1**, Table 8.9 reports an interesting subset of the results obtained for the **RQ2** variables.

Also in this second part of the experiment, we found that the interaction between the treatment and the laboratory has an impact on one of the monitored perceived efforts, i.e., the effort required in executing (matching or formulating) queries. The boxplot in Figure 8.6a shows that the perceived effort, in general, increases from the first to the second laboratory. However, by looking at the interaction plot in Figure 8.6b, it is clear that the described behaviour is determined by NL queries, while the perceived effort in formulating BPMN VQL queries decreases from the first to the second laboratory. These results suggest that the effort required by BPMN VQL could decrease by practice and encourage us to further investigate this aspect, for example by performing longer training sessions.

Finally, we found, as shown in boxplot in Figure 8.7, that the experience in using and modelling ontologies positively influences the perceived effort in understanding the



(a) Boxplots related to the perceived effort in matching queries in the first and the second laboratory



(b) Interaction plots of treatment and laboratory on the perceived effort in matching queries

Figure 8.5: Boxplots and interaction plots related to the perceived effort in matching queries

Laboratory (L)			
Variable	**TR**	**L**	**TR:L**
P_{QE}	**0.04115**	0.18766	0.42271
R_{QE}	**0.01048**	0.60672	0.60672
FM_{QE}	**0.003989**	0.308016	0.471126
$PEQE$	0.11426	0.28399	**0.03957**

Process Modelling Experience (PME)			
Variable	**TR**	**PME**	**TR:PME**
P_{QE}	**0.009031**	0.78749	0.92834
R_{QE}	**0.01050**	0.61042	0.61042
FM_{QE}	**0.004927**	0.586534	0.738287

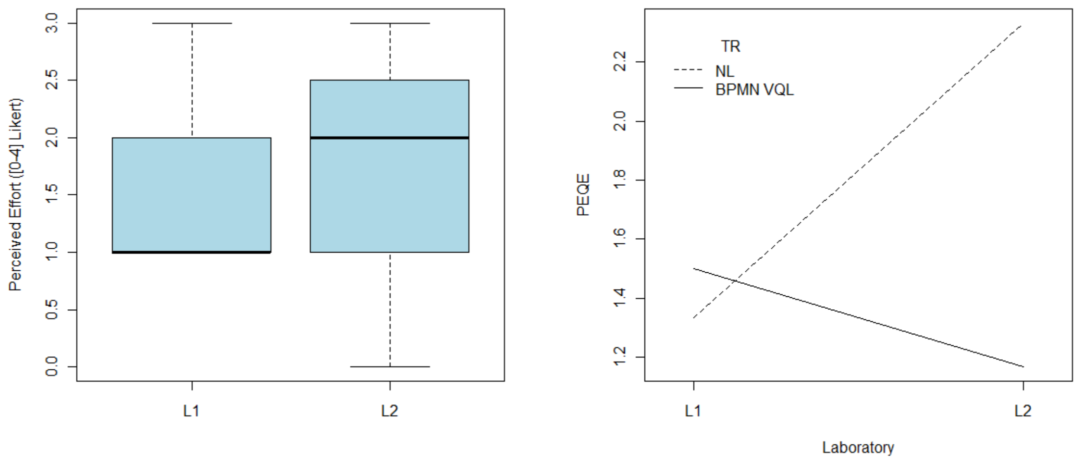
Object (O)			
Variable	**TR**	**O**	**TR:O**
P_{QE}	**0.02819**	0.05160	0.15480
R_{QE}	**0.009637**	0.466713	0.466713
FM_{QE}	**0.001803**	0.064023	0.121759

Ontology Experience (OE)			
Variable	**TR**	**OE**	**TR:OE**
P_{QE}	**0.04623**	0.31800	0.58522
R_{QE}	**0.009404**	0.43603	0.43603
FM_{QE}	**0.005262**	0.960006	0.832598
$PESU$	0.16655	**0.01406**	0.90934

Experience (E)			
Variable	**TR**	**E**	**TR:E**
P_{QE}	**0.0359**	0.26768	0.12192
R_{QE}	**0.01131**	0.87667	0.87667
FM_{QE}	**0.004453**	0.578383	0.419483

Visual Language Experience (VLE)			
Variable	**TR**	**VLE**	**TR:VLE**
P_{QE}	**0.0458**	0.2782	0.6524
R_{QE}	**0.01135**	0.90848	0.90848
FM_{QE}	**0.005168**	0.702525	0.987179

Table 8.9: Cofactor analysis related to **RQ2**



(a) Boxplots related to the perceived effort in executing queries in the first and the second laboratory

(b) Interaction plots of treatment and laboratory on the perceived effort in executing queries

Figure 8.6: Boxplots and interaction plots related to the perceived effort in executing queries

natural language specification used for describing tasks, i.e., the more the subject is expert in modelling ontologies, the less is the effort perceived in understanding the specification. This result can be traced back to the idea that a different mental attitude (e.g., matching queries rather than formulating them) can lead to different results and effort perception (as remarked in Section 8.4).

Survey Results

In order to be sure that everybody knew and clearly understood the basic concepts of BPMN and ontologies, explained in the training session, we evaluated the subjects' level of comprehension about these two topics, by asking them to answer some simple closed questions in the pre-questionnaire. We measured the degree of comprehension of BPMN as well as ontologies and ontology concepts by computing the percentage of correct answers. We found an average degree of comprehension of 1 for the BPMN and 0.96 for the ontology and the ontology concepts, thus clearly showing a good comprehension of the two topics.

In the post-questionnaire, besides the perceived effort, subjects were also asked to provide personal judgements on:

- their training and understanding of BPMN VQL;
- the BPMN VQL;

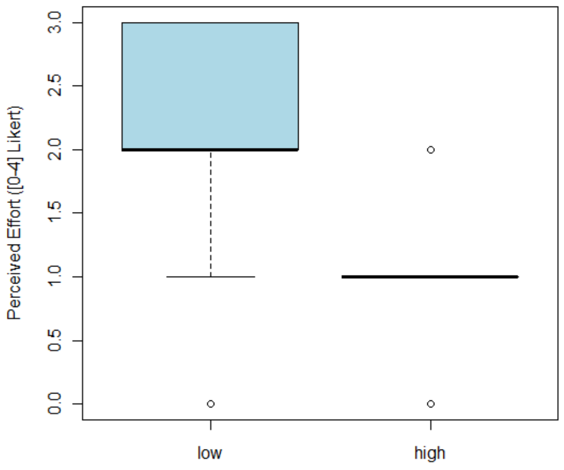


Figure 8.7: Boxplots related to the perceived effort in specification understanding for subject with low and high experience in modelling ontologies

- the utility of querying processes for supporting designers and analysts in understanding and maintaining processes;
- the benefits gained by BPMN VQL queries versus the effort associated with their formulation.

A summary and a description of the factors considered in the post-questionnaire is reported in Table 8.10.

For each of the factors, we performed a one-tailed unpaired (Mann-Whitney test) analysis in order to investigate:

- whether the subjects understood the BPMN VQL before starting the experiment;
- whether the subjects had a positive judgement about the BPMN VQL (in terms of the effort its use requires and expressive power);
- whether the subjects found querying processes for documentation and maintenance purposes useful; and
- whether the subjects judged the effort required by the BPMN VQL compensated by the benefits gained.

Factor	Description	mean	p-value	Cohen d
UVQL	BPMN VQL Comprehension	2.75	**0.001331**	**1.407125**
TEVQL	BPMN VQL Training Effectiveness	0.416666667	**3.444e-06**	**4.348508**
JEQU	Judgement on the effort required in understanding BPMN VQL queries	1.333333333	**0.002554**	**1.211060**
JEQF	Judgement on the effort required in formulating BPMN VQL queries	1.083333333	**3.644e-05**	**2.517557**
JEPVQL	Judgement on the expressive power of BPMN VQL	2.75	**0.0001164**	**2.345208**
JEQS	Judgement on the querying processes support to process understanding and maintenance	3.166666667	**1.228e-05**	**2.857738**
JVQLB	Judgement on the balance between benefits and limitations of the BPMN VQL	3.583333333	**3.444e-06**	**4.348508**

Table 8.10: Personal judgement survey results

The results of the analyses (reported in Table 8.10) show that the BPMN VQL has been understood by subjects (the understanding of the BPMN VQL declared by subjects ranges from “medium” to “very high”) and that the BPMN VQL training was effective (the BPMN VQL training has been declared by subjects either “very useful” or “useful”). In both cases the p-value was lower than 0.05 and the Cohen *d* effect size greater than 0.8.

Moreover, subjects did not judge the general effort required for understanding, as well as for formulating BPMN VQL queries, high. In detail, the declared efforts for both understanding and formulating BPMN VQL queries range from “very low” to “medium”, with the exception of one “high” judgement for the BPMN VQL understanding. The judgement provided by the same subject for the two different assignments (i.e., understanding or formulating queries) is the same in most of the cases, with few exceptions in which, surprisingly, the same subject judged formulating BPMN VQL easier than understanding them. Finally, though we are aware that subjects’ experience with BPMN VQL was limited, their intuition about its expressive power was positive: the formulated judgements ranged between “medium” and “high”.

A similar positive opinion is also expressed by subjects with respect to the usefulness for business process analysts and designers, when querying processes for documentation and maintenance purposes. In detail, all the subjects judged querying processes “useful” or “very useful” for documentation and maintenance purposes, with the exception of one subject considering the advantages provided by querying processes only “medium”. This confirms, though informally, our assumption that querying processes helps in understanding and documenting them.

Finally, subjects found the overall advantages deriving from the use of BPMN VQL queries for documentation and maintenance purposes superior to the possible drawbacks in terms of effort required in understanding and formulating BPMN VQL queries. All the subjects, in fact, evaluated the effort required in using BPMN VQL as "mostly" or "definitively" justified by the benefits gained by the use of the BPMN VQL itself (p-value=$3.444e-06$ and $d = 4.348508$).

8.3 Threats to Validity

8.3.1 Conclusion Validity

Conclusion validity deals with the relation between the treatment and the outcome.

In order to ensure such a validity, since not all the preconditions required by parametric statistical tests held in our study, we used non parametric tests (the Wilcoxon test) for our analysis of the main factor. ANOVA was instead used for the analysis of the cofactors. In fact, though it is parametric, it is a robust test and part of its results is also checked against the outcomes of the non parametric Wilcoxon test.

For the evaluation we chose to use both objective and subjective metrics. The first type, including metrics widely used in Information Retrieval (i.e., precision, recall and F-Measure) as well as the time spent to perform the assignments, provides a real and robust measurement of the performance of the two approaches. However, since our goal is to globally evaluate the ease of use of the approaches, we believe that also the subjective perception has to be taken into account. To this aim, we resorted to personal judgements about the effort required by the different treatments involved in the assignments. Though they are subjective measurements, by using standard settings and scales, we were able to apply statistical tests to the collected data.

8.3.2 Internal Validity

Internal validity threats concern external factors that could affect the dependent variables considered.

By performing an analysis of the possible cofactors (by means of ANOVA), we found that some of them have an influence on the dependent variables. The effect of the confounding factors, however, was limited by the adoption of the balanced design. In detail, we found that the laboratory session impacts the precision of the results and the time spent in the *Query Understandability* assignment. Moreover, the interaction between the treatment and the laboratory session has an influence on the perceived effort in query matching and query execution in the *Query Understandability* and *Query Execution* assignments, respectively. The influence of the laboratory session is mainly due to the limited time devoted to training subjects and hence to the need of more practice both with BPMN and with BPMN VQL. The impact of the laboratory session on the data is however mitigated by the balanced design and by the fact that the training session was the same for all the subjects, thus balancing and therefore limiting the effect of this factor.

Finally, though subjects were aware of the goal of the study they were not informed about the hypotheses.

8.3.3 Construct Validity

Construct validity is related to the relationship between theory and observation.

In order to limit the construct validity threats, we conducted a controlled experiment (i.e., laboratory sessions were performed under our supervision) and we carefully measured the analysed data. Precision, recall (and hence F-Measure) were measured in an objective way, by validating the answers provided by subjects or obtained by running the query formulated by subjects against the correct and a-priori known answers. Time was also measured in a precise way by providing subjects, during the experiment, with a watch providing granularity of seconds. In this way, they were able to report exact start and end times and hence the time spent was recovered as precisely as possible. Finally, though subjective, personal judgements too were measured by means of standard scales.

A possible construct validity threat is related to the type of tasks: they are not real-life tasks, but they have been conceived in order to be as similar as possible to real-world tasks.

The two compared approaches are quite different: the BPMN VQL one requires tool support, while the other does not. We tried to limit the effect of this threat by only simulating the use of the tool, thus choosing to penalize the BPMN VQL approach rather than conducting an unfair experiment and hence causing another threat to the construct validity. This solution, however, does not not prevent us from comparing the matching of NL queries and the formulation of BPMN VQL queries.

8.3.4 External Validity

External validity is related to the generalization of the findings. The number of subjects, and hence the number of points considered was not high (12 people) and subjects involved in the study are all PhD and researchers. Although their experience with process modelling, with the use of ontologies and with visual languages widely ranges (from less than 6 months to more than 5 years) we did not find any specific trend (except for the positive influence of the ontology experience on the perceived effort in understanding natural language specifications). This consideration encourages us in believing that the obtained results could be generalized also to experienced business designers and analysts.

A second threat to the external validity is related to process objects: they do not have the dimension of real life processes, due to the need to provide subjects with processes that can be managed in a reasonable time. However, we tried to mitigate this threat by choosing processes describing real-life procedures.

8.4 Discussion

The main goal of our analysis was investigating the difference of understanding as well as executing NL versus BPMN VQL queries, both from an objective point of view (in particular with respect to the accuracy of the results obtained by the query execution and to the time spent to perform the task) and a subjective point of view (in terms of perceived effort).

We found that, on average, the results obtained when manually matching BPMN VQL queries are more precise, more complete and, hence, more accurate than those obtained when manually matching NL queries. Though the result obtained for the recall only shows a trend that is not statistically relevant (i.e., the p-value of $R_{QU} > 0.05$), the one obtained for the precision clearly shows that BPMN VQL allows to be more precise in understanding requests related to semantically annotated processes. This finding could be explained with the higher formality of BPMN VQL with respect to NL, that hence better fits with the BPMN description, enriched with semantic annotations, of business processes.

Moreover, the positive result obtained for the accuracy of the answers is not excessively penalized by the effort required in terms of time. On average the time spent for performing the *Query Understandability* assignment in case of BPMN VQL queries is only 3.8% more than the time spent for performing the same assignment in case of NL queries (as also clearly shown in Figure 8.2b).

However, the additional time actually required for performing the understanding assignment with BPMN VQL queries rather than with natural language, is not perceived as an effort increase by subjects. As shown in Table 8.6, in fact, the effort perceived in BPMN VQL query understanding is lower (p-value < 0.05) than the one perceived for understanding NL queries. Moreover, we found that also the effort perceived in understanding the ontology, as well as in matching the query is significantly lower (p-value < 0.05) for BPMN VQL queries rather than for NL queries. The first result could be due to the formal structure in which ontologies are organized, which is closer to the BPMN VQL than to the NL. Similarly, we can speculate that having in mind a graphical representation of the pattern to look for, as well as having a clear and formal description of the semantics of the searched pattern components, could relieve the effort required by the matching task. By looking at Figure 8.2c we can also note that, when matching NL queries against the process, the most expensive aspect of the activity seems to be query understanding (on average 2.08 in the Likert scale, i.e., a perceived effort slightly higher than "medium"), followed by query matching (on average 1.83 in the $[0, 4]$ scale). This observation is in-line with the qualitative answers provided by subjects. The main difficulties found by subjects in understanding NL queries, in fact, are the ambiguity and the lack of precision of the natural language (reported by 6 subjects), as well as the difficulty in mapping natural language to structural properties of the process (5 subjects). When matching BPMN VQL queries, instead, the aspect requiring more effort is the actual query matching (on average 1.33 in the Likert scale), that is however lower, on average, than the effort required by the NL query match. In this case, the difficulties encountered by subjects in understanding queries are not common to more than one subject, and they differ from subject to subject.

By considering the collected results and taking into account the different factors analysed, we can hence reject the first null hypothesis and affirmatively answer the research question **RQ1**: understanding BPMN VQL queries is easier than understanding NL queries.

With respect to the research question **RQ2**, we found several statistically relevant results. In this case, in fact, the results obtained when automatically executing BPMN VQL queries formulated by subjects are not only more specific than those obtained by manually executing NL queries, but they are also more sensitive, and hence, more accurate.

Also in this case, the good result obtained in terms of accuracy is not heavily penalized by time performances. The time spent for formulating BPMN VQL queries, in fact, is, on average, just slightly higher than the one spent in matching NL queries (it took about 10% more of the time required for NL queries). However, the high average time of BPMN VQL queries is mainly due to two outliers. As shown in Table 8.5, in fact, the median value of the BPMN VQL query formulation time is lower than the median time required for matching NL queries (107.75 seconds versus 113.125 seconds). Moreover, since, when the process size increases, the time required for the BPMN VQL query formulation remains unchanged and the time spent for the automatic BPMN VQL query execution increases of the order of milliseconds (with respect to the almost null time considered in the study for the small *Bank Account Process* and *Mortgage Process* processes), the BPMN VQL approach scales better (with respect to the time spent) than the NL one, heavily influenced by the manual query matching time. For example, the time required for manually matching NL queries on the *Bank Account Process*, whose size (in terms of graphical elements) is about 80% of the *Mortgage Process* size, is 25% lower than the time spent by subjects for the same activity on the *Mortgage Process*.

Finally, we got other interesting findings related to the perceived effort in the *Query Execution* assignment. Though the results are not statistically significant, the effort perceived in formulating BPMN VQL queries is overall lower (on average 27% lower) than the effort required for matching NL queries, as shown in the boxplots in Figure 8.3c. Moreover, the perceived effort required for understanding natural language specifications of the tasks to perform seems to be positively influenced by the type of activity to be executed. In other words, understanding natural language specifications with the aim of transforming them into BPMN VQL queries is perceived as easier than understanding the same specifications with the aim of matching them against the process. Moreover, by inspecting the boxplots in Figure 8.3c, the same observation made for the *Query Understandability* assignment (i.e., understanding the natural language takes most of the effort involved) is confirmed also for the *Query Execution* assignment both for BPMN VQL and NL tasks. The perceived effort required for understanding natural language specifications is higher (with respect to the same Likert scale) than the effort perceived when formulating BPMN VQL queries or matching NL queries against the process.

The qualitative answers related to the main difficulties faced in formulating BPMN VQL queries for this assignment mainly concern the lack of the use of a tool for inspecting the ontology (reported by 3 subjects) and the poor experience with the BPMN VQL (reported by 3 subjects, too). These answers further encourage us in believing that, with the tool availability and with a bit more of practice, results could further improve.

By taking into account these observations, we can, hence, also provide an affirmative answer to **RQ2**.

As confirmed by the answers given to the last question in the post-questionnaire, we can conclude that, overall, the proposed BPMN VQL language, being easier to understand than natural language and making it easier to retrieve results scattered across processes, provides a good support to business designers and analysts in documenting as well as in retrieving specific information scattered across large processes, as for example crosscutting concerns.

Though a detailed evaluation has been limited to only one branch of the whole work (the BPMN VQL), the positive results we obtained make us confident taht we could get similar results also with other similar uses of the framework described in previous chapters (e.g., the visual representation of aspectized crosscutting concerns). We plan to investigate this in future works.

Chapter 9

Related Works

Hereafter we analyse relevant works in the literature related to the main topics presented in the thesis: reverse engineering of business processes and empirical investigations related to process understandability metrics (Section 9.1), semantic annotation of business processes (Section 9.2), constraint verification (Section 9.3), crosscutting concern management (Section 9.4), exception handling management (Section 9.5) and, finally, empirical studies involving visual process query languages (Section 9.6).

9.1 Reverse Engineering and Understandability Metrics

In this section we first provide an overview of the main works related to the reverse engineering of business processes (Subsection 9.1.1) and we then provide an overview about the main works relating process metrics and understandability (Subsection 9.1.2).

9.1.1 Reverse Engineering

Works in the literature for the reverse engineering of business processes from existing applications can be divided into static and dynamic approaches. While works in the first group statically analyse software artefacts, works in the latter dynamically examine the application execution.

Typical examples of static approaches are those recovering process models from the source code. For instance, Zou et al. [198] propose the use of static analysis of the source code to extract the business processes implemented by e-commerce applications. On the other hand, process mining techniques (e.g., [24], [178]), which try to infer processes by analysing the workflow logs containing information about process executions, represent typical examples of dynamic approaches. Though some works that are related to the recovery of organizations and roles (e.g., [177]) from execution logs or that are more focused on the recovery of interaction protocols between different participants (e.g., [124]) exist in the process mining field, most of the effort has been devoted to control flow mining. For instance, van der Aalst et al. [179] proposed the α-algorithm that detects causal dependencies (e.g., sequences, potential parallels, alternatives) among activities traced in logs by means of some heuristics. A Petri Net model of the mined process is then constructed according to the activity dependencies.

Moreover, focusing on Web applications, approaches in the literature can be classified according to the type of artefact they investigate to recover the process model.

Code-based Approaches. These techniques are focused on the analysis of the application source code. As mentioned above, for example Zou et al. [198] extract the business processes implemented by e-commerce applications by statically analysing the source code. In detail, the following steps are applied: (i) the source code is parsed to identify code entities candidate to be business entities; (ii) the source code is statically analysed to recover the control structure that manages these entities; (iii) the as-implemented workflow is built. A set of rules is applied to reduce the size of the analysed code and data (e.g., utilities and libraries are not considered).

GUI-based Approaches. These techniques are focused on the analysis of the application GUI without accessing application artefacts (such as the source code). For instance, Kim et al. [90] introduce an approach for recovering business processes by starting from the analysis of the forms contained in the GUI of an application. The intuition is that the GUI can be analysed to identify the business data: data in input and output to/from the application are business data. Kim et al. propose this technique in particular for business process reengineering.

GUI+code-based Approaches. These techniques recover the implemented process by starting from the analysis of the application GUI (mainly used for identifying business objects) and enrich it by exploring the application source code by means of static code analysis. For instance, Zou et al. [199] (and other works of the same authors such as [197]) describe an approach for recovering a two-layers process from a Web application. The high-level process is built by analysing the navigational structure of the application, while the low-level process details the high-level and it is generated by statically analysing the application code. Di Lucca et al. [52] introduce an approach for recovering business object models from applications. The approach focuses on the analysis of the elements of the application GUI (to identify the business objects) and on the use of a static analysis of the application code (to identify the relationships among business objects).

Data-based Approaches. These techniques are focused on the analysis of the data managed by the application. For instance, Paradauskas et al. [138] describe an approach to extract business knowledge from legacy code. The starting step of the approach is the analysis of the data stored in the database with the aim of recovering its schema. The schema is then semantically enhanced using clues extracted by analysing the system artefacts (e.g., source code, business reports, e-mail correspondence and corporate memos). Another example is the work by Hung et al. [81]. This work introduces an approach to extract business processes by the analysis of the communication flows between business-logic and business data of a three-layers Web application. To this purpose, a static code analysis is applied for controlling fetch and update operations used in the code to put data into and get data from the database.

Query-based Approaches. These techniques are focused on the analysis of several application artefacts (e.g., code, documentation) by means of (pattern-based) queries exploiting the presence of specific information in such artefacts. For instance, Ghose et al. [67] present an approach for querying several system artefacts (documentation, source code and web-content) with the aim of extracting its underlying process. Two types of artefacts are considered: *text* and *model.* The first one groups documents such as manuals, requirement documents, mission/vision statements, meeting minutes. Model artefacts are structural documentation of the software described by using notations such as UML and enterprise models.

Our reverse engineering technique is a GUI-focused approach that differs from the others in the literature since it extracts business processes exposed as Web applications by using a dynamic analysis in which application GUI elements are traced. No specific application knowledge and artefact availability are required. The reasons behind the choice of a dynamic analysis approach are mainly the following: (i) software artefacts are often not fully available (e.g., the code of third-party components is not available); (ii) the operations performed by a Web system are usually executed according to user actions (Web applications are event-based systems); (iii) Web applications involve dynamism (e.g., dynamic page construction) and reflection (e.g., dynamic DOM manipulation), which make them hard to analyse statically.

However, differently from process mining techniques, in which traces are provided with the execution environment, in our approach the GUI elements of Web applications, considered as implicitly conveying information about the underlying process, are chosen to be traced and execution logs are registered with this information.

Furthermore, we apply different types of clustering techniques (structural, logical and semantic) to the recovered process models in order to improve their readability and understandability by modularizing them. Process modularity and possible criteria for process modularization have been investigated in the literature from different perspectives (e.g.,[14], [183] and [189]). For example, Weber et al. [189] define the modularization of a sub-process in process aware information systems as a change pattern. Vanhatalo et al. [183] define SESE (single-entry-single-exit) fragments, i.e., blocks with one entry and one exit point, as good candidates for the modularization of process models. In a work made available on-line during the thesis writing, moreover, Reijers et al. [150], with the aim of identifying and providing guidelines for an effective modularization of processes into sub-processes, propose and investigate three different modularization criteria (block-structuredness, connectedness and label similarity), corresponding to the three clustering techniques proposed in our approach. According to their exploratory study the connectedness criteria seem to be the most promising candidates.

9.1.2 Process Understandability

Understandability is one of the major factors impacting the overall quality of models [123] and, in particular, of recovered models used to document or comprehend existing systems and organizations [154]. Mendling et al. [111] empirically highlight what makes a process model understandable. They reason about six factors influencing the process model understandability: personal factors (e.g., modeller skills), model characteristics (e.g., process structure), modelling purpose (e.g., documentation), model domain (e.g., hospital organization), modelling language (e.g., BPMN), and visual layout strategies (e.g., horizontal layout). Their results show that (i) personal factors, such as modeller skills and knowledge, and (ii) model characteristics, such as the process size, strongly impact process model understanding. In a more recent work, two of the same authors [148], focused on two out of the six factors investigated in their previous work (personal and model factors), by replicating the experiment with modelling experts. They found that, by keeping constant the size of models, two of the model factors they analysed (i.e., average connector degree and density) are related to the model understandability. Moreover, though results are not completely reliable due to the constant size of models, they found that personal factors have a stronger impact than model factors on the capability of understanding process models. Finally, their results also show that there is not so much difference between students' and practitioners' performance, though students more trained performed better than both less-trained students and experts. Other works, instead, focus on a single specific aspect potentially influencing process understandability. For example, Cardoso [29] investigates the impact of the control-flow complexity on the process model complexity perceived by subjects and he found that the two complexities are actually related. Reijers et al. [149] analyses the impact of modularization on process model understanding. Their analysis reveals that modularization in process models is positively related to its understandability and that this effect is more evident in large process models. Other works investigate the relation of process metrics with other factors. For example, Canfora et al. [27] focus on the relation between metrics of size, complexity and coupling with maintainability. Mendling [114] analyses the relation between structural metrics and error probability. Finally, only few works exist devoted at investigating the relation between a complete set of metrics and understandability of general process models (e.g., [110] and [55]). For example, Rolón et al. [55] found that only 12 out of the 29 metrics they analysed are related to process model understandability.

Hence, summarizing, existing works in the literature focus on: the identification of factors making process models understandable (e.g., personal factors, model characteristics) [111, 148]; the analysis of specific factors impacting understandability (e.g., control-flow complexity [28] and process modularization [149]); the relation between process metrics and factors as maintainability or errors in process models (e.g., [27] and [114]); the evaluation of the understandability of generic process models (e.g., [110] and [55]). Differently from all these approaches, our empirical investigation aims at identifying a set of metrics to be used as indicators for the understandability of process models recovered from existing applications, that, up to now, is missing.

9.2 Business Process Semantic Annotation

In the literature several approaches have been proposed aimed at providing a shareable and understandable basis for business modelling by attempting to integrate different cross-domain and cross-organizational aspects. Some of them try to merge different perspectives in new languages (e.g., [94]); others, instead aim at adding missing semantic information.

The problem of adding formal semantics to business processes has been extensively investigated in the literature [16, 41, 53, 54, 71, 93, 102, 167, 190, 195]. We can roughly divide the existing proposals into two groups: (1) those adding semantics to specify the dynamic behaviour exhibited by a business process [190, 195, 93], and (2) those adding semantics to specify the meaning of the entities of a process in order to improve the automation of business process management [102, 16, 54, 167, 41, 71]. Our approach belongs to the second group.

Thomas and Fellmann [167] consider the problem of augmenting EPC (Event-Driven Process Chain) process models with semantic annotations. They propose a framework which joins process model and ontology by means of properties (such as the "semantic type" of a process element). Markovic [109] considers the problem of querying and reasoning on business process models. He presents a framework for describing business processes which integrates functional, behavioural, organizational and informational perspectives: the elements of the process are represented as instances of an ontology describing the process behaviour (based on π-calculus), and the annotations of these elements with respect to the ontologies formalizing the aforementioned perspectives are described as relation instances. Born et al. [22] propose to link the elements of a business process to the elements of an ontology describing objects, states, transitions, and actions. These proposals differ substantially from ours, which establishes a set of subsumption (aka subclass or is_a) relations between the classes of the two ontologies being integrated (BPMN metamodel and domain ontology), instead of associating annotation properties to the process instances. De Nicola et al. [41] propose an abstract language (BPAL) that bridges the gap between high-level process descriptions (e.g., in BPMN) and executable specifications (e.g., in BPEL). The formal semantics offered by BPAL refers to notions such as activity and decision. They developed a BPAL platform that exploits reasoning to verify the compliance of process models with respect to a metamodel as well as of process traces with respect to the process definition [42]. Differently from our approach, they have not yet integrated business ontologies in their frameworl and their language is based on Horn clause logic and their engine exploits Prolog systems. In the SUPER project [54], the SUPER ontology is used for the creation of semantic annotations of both BPMN and EPC process models in order to support automated composition, mediation and execution. Recently, Groener and Staab [71] presented a pattern-oriented approach in which OWL representation and reasoning capabilities enable expressive process modelling and retrieval. Their process formalisation considers the language primitives of the UML-Activity Diagram and the connection with the domain knowledge involves the representation of terminological information about activities and subactivities only. Di Noia *et al.* [130] semantically annotate building blocks (BBs), i.e., flexible and transparent pieces of functionality within ERP (Enterprise Resource Planning) systems. By exploiting standard and non-standard reasoning services, they provide a framework that automatically selects the set of BBs needed for satisfying a requested business process and, if it does not exist, provides explanations on what is in conflict and what is still missing to cover the request. Lin [102] in her work uses the semantic annotation of process and goal models with the purpose of guaranteeing the interoperability of process and goal models. She exploits a GPO (General Process Ontology) ontology to reconcile the heterogeneous semantics of process modelling constructs of different process modelling languages, while she uses concepts belonging to an agreed domain ontology for reconciling model contents.

Our work represents an extension of the existing literature in that, semantically annotating BPMN process elements with concepts of a domain ontology, aims at supporting business experts with different services (e.g., automatic verification of constraints and process querying).

Moreover, in our approach, we also support process designers in performing the time-consuming task of process annotation.

Several tools exist for the automatic and semi-automatic semantic annotation of Web documents and Web Services in the Semantic Web field. According to a classification based on the type of automation they exploit, we can identify three main groups [172]. Tools belonging to the first group (e.g., KIM [142] and AeroDAML [91]) use rules capturing known patterns for the automatic annotation. Works belonging to the remaining two groups are instead based on learning systems: some of them (e.g., MnM [184] and Melita [34]) are supervised systems, i.e., they learn from user annotations, while others apply unsupervised learning (e.g., Armadillo [33]), i.e., they employ different techniques in order to avoid consequences deriving from wrong manual annotations. Moreover, most of these tools apply some form of Natural Language Processing (NLP). Our approach exploits techniques from NLP, as for example word stemming, grammatical category classification as well as POS (Part of the Sentence) recognition. However, these techniques are applied to short sentences rather than complete text sentences.

Though some work also exist in the literature for the semantic annotation of Web services starting from the structured information provided by their WSDL (e.g., [139]), few work has been done in the specific field of (semi-)automatic semantic annotation of business processes. In their work, for example, Bögl et al. [21] target EPC process model elements. In detail, in order to provide an automated semantic annotation of EPC functions and events, they analyse the textual structure of natural language labels of EPC elements by means of semantic patterns and relate them to instances of a reference ontology.

Wang et al. [188] propose, instead a weighted mean of three similarity measures (syntactic, linguistic and structural), based on string-matching, for suggesting domain annotations in supply chain models. In detail, in their work, they consider the SCOR (Supply-Chain Operation Reference) ontology (an ontology developed to specify constructs and terminology in supply chain processes on the basis of the SCOR model), as the domain ontology and a BPMN ontology for the process structural knowledge. By combining these two ontologies, a so-called scorBPMN ontology is derived, which is used to suggest annotations for BPMN process elements, by ranking candidate annotations from scorBPMN according to their weighted similarity measure with BPMN process elements.

Born et al. [22] also deal with the problem of supporting process designers in the integration of domain ontology and BPMN process modelling knowledge. The process information they exploit to this purpose is related to the process structure and the matching technique they use is mainly based on string matching (e.g., distance metrics). Moreover, they integrate their approach for semantic annotation suggestions in the Maestro for BPMN, a modelling tool from SAP Research and use it for web service discovery and composition [23].

Differently from these two latter techniques, our approach takes advantage of linguistic analysis (natural language parsing) of process element labels and of concept names, by looking for special semantic patterns, for BPMN activities, similar to those proposed by Bögl et al. [21] for function and event EPC elements. Moreover, our approach differs from all those described above because it exploits a measure of information content similarity for providing suggestions to business designers. The same similarity measure is also used for supporting the business designer in the disambiguation of the domain ontology with respect to the possible senses of each concept (thus reducing the search space of the automatic suggestion algorithm) and in the ontology extension and/or creation (when necessary).

9.3 Constraint Verification on Business Processes

There exists a number of works realizing constraint verification in business processes in the context of business process compliance. Compliance checking has been defined by Governatori et al. [70] as the adherence of one set of rules (*source rules*) to another set of rules (*target rules*). By looking at the mere verification aspect of process compliance works, we can place our approach in this trend of research.

In general, we can classify process compliance checking approaches in two main groups: those realizing compliance checking backward and those realizing compliance checking forward. Backward techniques are reactive approaches, i.e., they can only detect non-compliance by looking at already executed process instances, but they are unable to prevent non-compliant behaviours. Forward techniques, instead, are pro-active approaches.

By targeting the verification of rules during design time or execution time, they can, in principle, allow to prevent (in case of design-time) or solve (in case of run-time) the problem.

Backward Approaches

Backward techniques verify if executions of business processes are in accordance with certain constraints or rules. These works often use traces as representatives of process instances. An example of this kind of approaches is the work by Rozinat et al. [156], in which two metrics (fitness and appropriateness, quantifying model completeness and generalization, respectively) are used for measuring the adherence of model behaviours with execution trace behaviours. The ProM[1] LTL Checker by Van der Aalst et al. [176], instead, uses a variant of LTL that supports absolute time, while avoiding state explosion. Another work in this group is the SCIFF/CLIMB framework [32, 121] by Chesani et al. It is based on Abductive Logic Programming and it performs compliance checking between process execution traces and rules specified in declarative languages. In detail, SCIFF is able to formalize ConDec [140] constraints, while CLIMB uses extensions of Logic Programming for modelling and verifying business processes, extending the expressiveness of ConDec. Horn clauses are instead used by De Nicola et al. [42] to describe process models and traces in BPAL and to verify the compliance of process traces with the process model by exploiting Prolog systems.

Forward Approaches

Run-time Approaches. Run-time forward techniques target executable business process models and potentially allow to solve non-compliance problems in time. An example of this type of approaches is the one proposed by Namiri et al. [128]. Constraints (here called controls) are described as declarative rules external to the process. Their monitoring at run-time, however, requires the manual selection of the concrete control pattern. Similarly, Weber *et al.* [190] introduce a notion of Semantic Business Process Validation (SBPV), which exploits semantic annotations to verify constraints about the process execution semantics. In their work, semantic annotations referring to a background ontology are used to ensure that an executable process model behaves as expected in terms of preconditions to be fulfilled for the execution and its effects. Approaches based on LTL for specifying run-time requirements, instead, have to face the problem deriving from the fact that standard models of linear temporal logic are infinite traces, while, during the

[1]`http://prom.win.tue.nl/tools/prom/`

execution, traces are finite and new events can occur. In order to deal with this problem Bauer et al. [15], for example, introduce a 3-values semantics (true, false, inconclusive) for LTL formulas. Another example of run-time compliance checking technique is the SCIFF/CLIMB framework [6, 121]. The framework, in fact, allows not only to check whether a complete execution trace complies with a given rule, but also to dynamically reason on a partial trace, by exploiting a list of pending expectations.

Design-time Approaches. Design-time techniques for checking the compliance aim at guaranteeing that all process instances will be compliant to a set of regulations. Some of these approaches are conceived to be applied during the modelling phase, while others use techniques like model checking to verify properties in already designed processes.

Some of the works in this group are based on the notions (derived from the normative field) of obligations, permissions and prohibitions investigated by Deontic Logic[2]. For example, Governatori et al. propose the Formal Contract Language [69] (FCL) for describing normative rules (called *business contracts*). FCL is a formalism combining Deontic Logic with logic of violations and thus allowing to represent exceptions as well as to capture violations, obligations resulting from violations and reparations. In detail, compliance checking is given by the notions of ideal, sub-ideal, non-ideal and irrelevant situations (*Ideal Semantics*), describing various degrees of compliance between execution paths and FCL constraints. Goedertier et al. [68], instead, introduce a language, PENELOPE (Process ENtailment from the ELicitation of Obligations and PErmissions), that allows to specify obligations and permissions (temporal deontic assignments) extracted from business regulations in order to generate a compliant process model to be used for verification and validation purposes.

The SCIFF/CLIMB framework [6, 121] also allows the static verification of process compliance. To this purpose, *g-SCIFF*, a generative version of the framework, is proposed. g-SCIFF is able to simulate execution traces starting from a given goal and to abduce event occurrences. In this way it is used to prove system properties at design time, or to generate counterexamples of properties that do not hold. Similarly, the BPAL framework by De Nicola et al. [42], can be used to verify the compliance of process models with respect to a meta-model (well-formedness). The process model, translated into a set of BPAL facts, and the metamodel, also described as a set of BPAL composition rules, constraints and inclusion axioms, are provided as input to the BPAL engine that exploits an extension of the Prolog system to check the consistency of the metamodel with respect to the process model.

[2]A definition of deontic logic can be found at `http://plato.stanford.edu/entries/logic-deontic/`

Other works express constraints as LTL formulas and use model checking techniques for verifying their compliance with process models. In order to deal with the complexity of these formulas, most of these works also propose a visual representation of constraints, close to the language used for describing the process control flow. For example, Forster et al. [59] use UML Activity Diagrams to specify the business process and PPSL (Process Pattern Specification Language), a graphical language similar to the Activity Diagrams, to represent the business rules. In detail, they transform the process from the UML Activity Diagram into a Labeled Transition System and PPSL rules into past-LTL formulas. A similar approach for processes described in BPMN is proposed by Awad et al. [10]. They adapt the visual query language BPMN-Q [9] to express the constraints they want to verify on BPMN processes. BPMN-Q queries are first used to extract sub-graphs from a BPMN process model repository and, then, converted into past-LTL formulas. The retrieved sub-graphs are reduced by removing BPMN elements non-relevant for the business rule and, once the state space has been adequately reduced (thus solving the problem of the state explosion), they are transformed in Petri Nets on which LTL formulas are verified by model checkers. Another approach of the same type but investigating rule verification on BPEL processes is the one proposed by Liu et al. [103]. They also describe business rules by means of a graphical language, BPSL (Business Property Specification Language) and then convert them into LTL formulas. The BPEL process, instead, is first transformed into a representation based on pi-calculus and then into a Finite State Machine; model checkers are used to verify the compliance of the rules.

A different example of design-time approaches is the work by Schmidt et al. [159], that is based on ontologies and exploits ontology reasoners for verification purposes. In detail, the authors define two ontologies (described in OWL): a process ontology and a compliance ontology. The process ontology contains the concepts needed to represent service processes and its classes are instantiated in order to provide an ontology-based representation of a process model. The compliance ontology, instead, contains concepts used to represent objectives and requirements of compliance rules. Some of these requirements (named syntactic requirements) can be directly encoded as OWL axioms, while the others require human intervention either to add missing concepts and formalize constraints to be verified by reasoners (semantic requirements) or to manually check the constraint (in case of pragmatic requirements).

Our approach belongs to the category of works verifying compliance at design-time. Similarly to the approach by Schmidt et al., it also populates the classes of a BPMNO ontology with instances of BPMN BPDs and it exploits ontology reasoners for checking constraints, thus allowing not only the detection of exceptions, but also their separate management as aspects and eventually further verification of the woven process model. However, our approach is theoretically grounded on Description Logics and applied to semantically annotated business processes. Hence, by paying the cost of the process semantic annotation with domain concepts, it does not require the manual addition of special concepts for representing semantic requirements. The formalization of constraints into Description Logic axioms and the availability of Semantic Web technologies allow us to automatically express and verify both syntactical and semantic constraints. Moreover, the patterns we identified allow to describe the main constraints related to the execution flow, shared with other languages as ConDec [140], CLIMB [121], BPMN-Q [10], PPSL [59] (e.g., existence, precedence and response), while introducing new types of requirements. Some of these new requirements are specific of the BPMN as, for example, those related to message flows; others, instead, are more general, like, for example, the existence of at least a path between two BPD elements.

9.4 Crosscutting Concerns

In this section we first provide an overview of crosscutting concerns in generic software (Subsection 9.4.1) and then we look at the approaches that apply techniques for the separation of concerns to processes (Subsection 9.4.2).

9.4.1 Crosscutting Concerns in Software

Some of the most challenging problems in software engineering are related to code understandability, maintainability, evolution and reuse. At the core of software engineering is the “divide and conquer” principle: a complex problem can be solved by breaking it up into smaller subproblems, that can be solved in isolation and combined modularly with each other to solve the original problem. This principle provides the basis for software modularization. However, existing programming paradigms fail to support proper modularization for scattered and tangled functionalities, commonly referred to as crosscutting concerns.

In order to face the problem of the separation of concerns, the AOSD community proposed several approaches, with different levels of invasiveness in the primary code, aimed at supporting the programmer in crosscutting concern documentation and browsing, in crosscutting concern mining, and in aspect refactoring.

Crosscutting Concern Browsing. Software maintenance and evolution often imply the need for localizing specific concerns. In large systems, this task is not trivial, especially for those concerns which crosscut the system. Several tools have been introduced in order to support the programmer in source code navigation and concern localization. Usually they allow users to browse the concern code starting from a "seed" of the concern (one or more source code entities tightly related to the concern) and incrementally extending the exploration to the whole concern implementation. The seed expansion can be realized semi-automatically, by proposing links to code related to the concern, or manually, by providing the user with a query language supporting source code navigation. Examples of such tools are JQuery and FEAT.

JQuery [85] is a generic source code browser developed as an Eclipse plugin. Eclipse is an IDE (Integrated Development Environment) which, by itself, allows to browse the source code looking for predefined, non customizable and mostly local code structures. JQuery extends Eclipse by allowing to logically query the code, so as to obtain results related to specific concerns (both crosscutting and non crosscutting). In the displayed results, more than one view coexist in the same visual space, thus facilitating inspection of multiple views at the same time. JQuery supports also refinement of previously defined queries, by means of filters, and execution of further queries on elements of an existing view, thus avoiding loss of context. JQuery is mainly focused on the exploration of the code, in that it represents the history of the exploration process, without providing explicit support for capturing the representation of a concern.

The FEAT [152] tool, on the contrary, is more specifically concern-oriented. It allows to explicitly model a concern as a container of source elements (classes, methods or attributes) and to graphically display a tree representation of the code contributing to its implementation. Starting from a "seed" of the concern, the full concern is incrementally built by querying the structural relationships in the source code.

Crosscutting Concern Mining. In large systems, even with browsing and documenting tool support, manual search for crosscutting concerns can be a difficult and error-prone task. A number of different techniques, proposed in the context of migration of legacy code to aspect-oriented code, have been developed to automate aspect mining. They can be grouped into static techniques, looking for candidate aspects in the source code, or dynamic, finding patterns in the executions (e.g., in the execution traces). Moreover, they can be based on formal analysis, like Formal Concept Analysis, on metrics (as the fan-in technique) or on heuristics.

For the purpose of aspect mining, FCA (Formal Concept Analysis) has been used in several different ways. FCA produces a lattice of concepts out of a relationship between objects and attributes. Concepts group maximal sets of objects sharing maximal sets of attributes. Tonella and Ceccato [170] applied the FCA algorithm to execution traces. They use execution traces associated with use cases as objects and methods invoked during the execution of the use case as attributes. Then, they focus on concepts containing traces belonging to a single use case and among these, they consider as candidate aspects those labelled by methods belonging to more than one module and to modules whose methods label more than one use case (thus enforcing scattering and tangling of the concern).

Tourwè and Mens [171] applied FCA to the static code, performing an identifier analysis. They consider modules and operational units as objects and meaningful substrings of their identifiers as attributes. They finally obtain maximal groups of classes and methods that share a maximal number of subterms in their identifiers.

Another work in which FCA is used for detecting crosscutting concerns has been proposed by Tonella and Antoniol [169]. They applied this technique in order to find design patterns. They consider groups of classes as objects and class relationships and properties as attributes of concepts. They finally find structural patterns without using any predefined library.

Aspect Refactoring. Software refactoring [61] is a technique that helps to improve the internal structure of a software system, while preserving its external behaviour. Improvements consist of design level enhancements. Their goal is to get a better organized, more readable and clean code, avoiding code duplications and producing a modularization easier to understand and maintain. Aspect refactoring techniques are based on the transformation of crosscutting concerns, either manually determined with browsing tool support, or automatically discovered by means of aspect mining tools, into actual aspects. This process implies a range of design choices for the right crosscutting concern organization and modularization into aspects, hence the introduction of new specific refactorings, as for instance those related to the advice choice. Several attempts aimed at organizing aspect refactorings into a coherent catalogue have been made, mainly for object oriented systems and with different levels of granularity. Monteiro [122], for example, proposes a low-level aspect refactoring catalogue for the migration from Java to AspectJ. At a higher level of granularity, Laddad [97] suggests some concrete applications of aspect refactoring, like aspect modularization of the logging functionality, of business rules, exception handling and design patterns.

9.4.2 Crosscutting concerns and processes

Although the idea of the separation of concerns originated in the context of general-purpose programming languages and was initially applied to the implementation phase only, its principles have been extended not only throughout the software development process (e.g. starting with aspect-oriented requirement analysis), but also to specific process languages.

Mezini and Charfi [7], for example, stress the lack of flexibility and modularization of crosscutting concerns in classic process definition languages. In their work, they propose to apply an aspect oriented approach to business processes and, in particular, to the associated process execution languages. AO4BPEL [7] is an aspect oriented extension of BPEL [39], designed to be as close as possible to the AOP programming language AspectJ [97], thus providing similar concepts for describing pointcut designators, joinpoints and advices. In AO4BPEL, advices are fragments of BPEL code, while pointcut designators take advantage of XPath expressions to locate the places in the BPEL process where the aspect is applied. The language allows to break the ”tyranny of the hierarchical decomposition” usually adopted in process definition languages, thus enabling a concern-based modularization. Such decomposition not only separates non functional crosscutting concerns, but, taking advantage of the particular structure of workflow processes, it also makes the system more open and adaptable to functional changes, by encapsulating functional concerns too. Dynamic weaving is realized by means of a custom BPEL engine.

Courbis and Finkelstein [38] propose an approach similar to AO4BPEL: XPath as pointcut language and a custom engine for dynamic weaving. The main differences lie in the choices related to the advice language, that is Java, and the crosscutting concern type, basically non functional.

Verheecke, Cibràn and Jonckers [186] present aspects as solutions for capturing concerns that are both typical of the web service world, like, for example, service selection and billing, and classic non functional concerns, e.g., transactions. They propose WSML (Web Service Management Layer), a layer that uses aspects implemented in JAsCo, a dynamic aspect oriented extension of Java, for representing aspects related to services and independent from the composition of services in processes.

Kongdenfha et al. [92], instead, use run-time weaving of aspects for adaptation purposes. In detail, they focus on the problem of service mismatching with external specifications (i.e., the external descriptions of the service interfaces). They classify possible mismatches (e.g., signature, order mismatch) according to a taxonomy and they propose aspect templates for dealing with each of these situations. Their joinpoint definition is based on a query language, similar to executable process query languages, but enriched with the capability to specify runtime conditions, while advices are expressed in BPEL.

On the other hand, Padus [26] focuses more on aspects concerning activity composition, i.e., on process centered aspects. Padus is yet another aspect-oriented BPEL extension, quite similar to AO4BPEL, but without dynamic weaving (so that it is independent from the BPEL engine used) and with some improvements. In detail, the proposed pointcut language is more abstract than XPath (so that it is possible to achieve independence from the document structure), the joinpoint model is richer (by allowing to capture all types of activities) and finally an explicit construct for the deployment is introduced in order to be able to specify instances of a given process by means of a logic language.

Finally, other works, similar to these approaches but not explicitly using aspects, exist in the literature. For example, Casati et al. [30] propose an approach for the management of flexible workflows based on rules and patterns to be used in particular for the separate modelling of exceptional flows. In fact, rules allow to model exceptional flows independently from the main flow, while patterns provide support to designers addressing exception and exception handling situations. A rule, in the WIDE framework they propose, is composed of an event (specifying when the rule is triggered), a condition (specifying a condition to be verified to activate the rule) and the action (the operations to be performed). Similarly to aspect definitions, their rules answer the when (event) and what (action) question, as well as allow to specify a condition to be verified.

All these works, however, mainly focus on the developers' perspective, without considering the business designers' one. In practice, business experts prefer a higher level modelling notation, such as BPMN to a process executable language. Our aspect-based language BPMN VRL, by proposing a solution for the modularization of crosscutting concerns into aspects at design-time and a syntax as close as possible to BPMN, takes into account the modellers' needs. In particular, similarly to lower level approaches (e.g., implementation-oriented), BPMN VRL has been applied for the separate management of exception handling mechanisms in business processes. Similar approaches, i.e., proposing the separate modelling of exceptional flows by exploiting rules, have been investigated in the literature.

In some cases, however, the explicit decomposition of the process into totally separate aspects may hinder, instead of simplifying, process design and comprehension for business experts. Hence, offering the possibility to choose among different degrees of modularization, similarly to the solutions proposed for the separation of concerns in general purpose code, can represent an important support for business designers. For example, a valid alternative to the decomposition into totally separate aspects is represented by crosscutting concern retrieval (mining) and documentation.

By taking inspiration from the aspect mining techniques described in the previous subsection, FCA has been applied to semantically annotated processes for automatically mining crosscutting concerns. In detail, maximal set of process elements, associated with a maximal set of semantic concepts (including superconcepts) they instantiate, are identified.

BPMN VQL allows not only to manually retrieve crosscutting concerns but also to query the process with respect to generic concerns, thus supporting designers in process browsing. Several languages for querying process models exist in the literature (e.g., [120, 119, 16, 9]). Some of them are textual, while others are visual languages. For example, BPQL [120], a textual language based on the Stack Based Query Language (SBQL) [164], mainly used to retrieve (and manage) information with specific characteristics related not only to the process structure, but also to execution objects and performers, belongs to the first class.

Some of these textual languages share with our approach the capability to query both structural and semantic knowledge. For example, Missikoff et al. [119] propose a language, similar to SQL (Structured Query Language), allowing to query the process also about business concepts and process traces, i.e., beyond the structural, both the business semantic and the execution dimensions are considered.

To the best of our knowledge, instead, only two process query languages have a graphical syntax: BPMN-Q [9] and BP-QL [17].

BPMN-Q [9] is a BPMN extension for visually querying business processes, differing from BPMN VQL on purpose and operators provided to the business designers. Its objective, in fact, is to query process repositories for retrieving process models with desired structural features, thus promoting process reuse.

BP-QL [17], instead, is a language for querying BPEL processes based on business process patterns, that allow to describe the desired control-flow or data flow patterns of interest. It enables the navigation along two axes (the *path-based* and the *zoom-in* axis), thus allowing users to have paths in query results, as well as to control the granularity in business processes. The BP-QL implementation exploits the graph matching functionality of XML and is based on Active XML (AXML)[3], an XML enriched with service calls to Web services. Also BP-QL, differently from BPMN VQL, is mainly conceived for querying repositories. Moreover, it targets executable processes described in BPEL [39] and it does

[3]`http://activexml.net`

not allow to query processes about semantic aspects,

9.5 Business Process Exception Handling

Among the requirements business designers are interested to ensure, exception handling holds a crucial role for enhancing the process robustness since the modelling phase. With the aim of supporting process designers in the correct management of exceptions, several works in the literature have investigated exceptions and their handling. Some of them classify exception handlers according to patterns at different levels of abstraction [30, 100, 157]; others [18, 43] exploit semantic information stored in a repository or added to the process in order to warn designers against potential errors and/or to suggest possible solutions for their management.

In the first group, Russel et al. [157], for example, propose a categorization of exceptions and exception handling in Process Aware Information Systems (PAIS). In detail, they classify exceptions into five categories (activity failure, deadline expiration, resource unavailability, external trigger and constraint violation) and exception handling on the basis of three different levels of granularity (activity, case and recovery action perspective). In case of the activity perspective, they analyse a rich description of the activity life cycle and try to identify possible exception handling reactions, according to the presence of non-normative transitions; with respect to the case level, they categorize the possible impacts of the exception on the other activities of the current instance or of other instances of the same process currently executing (continue workflow, remove current, remove all); and, finally for the recovery action they distinguish among three different strategies (no action, rollback and compensation). Their work of classification results in the identification of process exception handling patterns, obtained by considering all the possible combinations of these four factors. For each type of exception, they hence combine the different exception handling strategies at activity level (according to the location of the occurrence of the exception), case level and with respect to the recovery action. Exception patterns proposed by Lerner et al. [100], instead, are at a higher level of abstraction than those proposed by Russel et al.. In detail, they classify the patterns in three main categories: patterns allowing to choose among alternatives, patterns adding new behaviour and patterns cancelling some behaviour. Moreover, they provide evidence of pattern occurrence in real processes and a description of the pattern applicability (i.e., what problem can be solved with the specific pattern).

In the second group of works Dellarocas and Klein [43] propose the use of a reusable and extensible body of knowledge describing and classifying exceptions and their handlers for detecting, diagnosing and resolving exceptions. Eliahu and Elhadad [18] infer the existence of likely errors by analysing the structure of the process and some semantic information added to process activities in the form of semantic tags.

In our work we exploit semantics both for verifying the correct handling of exceptions and for modularizing their management into aspects, thus allowing not only the detection of the exception, but also the separate management and eventually further verification of the woven aspect. As already mentioned in the previous subsection, the use of aspects for exception handling has been deeply investigated in the AOP literature, for example for the AOP refactoring of object oriented code [97].

9.6 Visual Process Query Language Evaluation

Due to the relatively new introduction of visual query languages for business processes, no empirical evaluation on their usability exists in the literature.

We can identify, however, two main group of works (experiments about visual query languages and empirical studies about BPMN processes), which can be related to the BPMN VQL empirical evaluation.

Some works investigating the advantages of visual languages for querying databases rather than using standard textual languages, as SQL, can be found in the literature. For example, Catarci et al. [31] in their work conduct an empirical study with subjects for comparing the QBD* (Query By Diagram) visual query language, which is based on a conceptual data model, with the SQL language. Though they found that the effectiveness of the language varies depending on the types of queries and users, the general trend is in favour of the visual language. Sadanandan et al. [158], instead, conducted an exploratory study in order to investigate the usability of the visual language they propose for querying ontologies. Results confirm a high usability of the approach.

A second group of related works is the one empirically assessing the usability of the BPMN notation, which represents the basis of the BPMN VQL language. Recker et al. [145, 143] analysed BPMN against the Bunge-Wand-and-Weber ontology and validated their findings by means of exploratory studies. Though they identified some weaknesses in the language, they confirmed BPMN to be a mature language for modelling business processes. Recker et al. [146] evaluated BPMN versus EPC from a teaching perspective. Untrained BPMN modellers overperformed with respect to trained participants working with EPC. In a recent empirical study with human subjects about the usability of BPMN and UML Activity Diagrams, Birkmeier et al. [19] found that UML Activity Diagrams are at least as usable as BPMN. This result confirms the one obtained by Recker et al. [147] about BPMN complexity. In their study they found that BPMN has very high levels of complexity, but that, however, such complexity could be significantly reduced through the use of modelling conventions and limiting the use of BPMN symbols to a subset [125, 144].

Chapter 10

Conclusions and Future Works

The core of this thesis is the presentation of a framework that formalizes business process models enriched with semantic annotations into a knowledge base. The semantic annotation of business process models, besides clarifying their semantics, thus making them more understandable to people, enables several advanced analyses and manipulations, e.g., the documentation of crosscutting concerns by means of a visual and formal language (BPMN VQL), their semi-automatic mining in process models, as well as their aspectization, by means of another visual language (BPMN VRL). Moreover, the formalization into a knowledge base allows to automatically verify constraints and query semantically annotated process models, by enabling reasoning services. Since process models are not always available and, when they exist, enriching them with semantic annotations is an expensive task, we support designers and analysts with techniques for the reverse engineering of process models and for the automated suggestion of semantic annotations for business process elements. A preliminary empirical investigation of process structural metrics as early indicators of the recovered process model understandability and an empirical study with subjects investigating benefits of and efforts required by the BPMN VQL with respect to the natural language have been conducted.

In summary, this thesis contributes to the state of the art by:

- proposing a technique for the reverse engineering of business process models;
- investigating the use of process metrics as early indicators of the recovered process model quality (in terms of human understandability);
- presenting semi-automatic techniques supporting business designers in the semantic annotation of business process models with domain ontology concepts, as well as in the domain ontology construction and extension;

- introducing a business process knowledge base allowing to formalize both the structural and the business domain information of semantically annotated business process models;

- proposing an approach for constraint definition and automatic verification in semantically annotated process models;

- defining a visual language (BPMN VQL) to query business process models and document scattered and tangled business concerns;

- proposing a technique (based on Formal Concept Analysis) for the semi-automatic retrieval and documentation of crosscutting concerns in semantically annotated business processes;

- defining an aspect-oriented language (BPMN VRL) to modularize crosscutting concerns (e.g., exceptional flows) in process models;

- presenting the results of an empirical study with human subjects conducted to evaluate and assess the ease of use of BPMN VQL with respect to using natural language.

Despite the limitations of the work in this thesis, we believe that it shows how enriching process models with semantics and formality, while preserving their ease of use, can enhance the current support to business experts (hence strengthening the role of business process models as artefacts intended to be used by humans). On one side, in fact, results related to the quality of reverse engineered process models from Web applications and of semantic annotation suggestions show that, even if applications are not documented, process models describing their flow can be recovered and that business experts can be partially relieved from the time-consuming activity of semantically annotating processes. On the other side, results related to the performance of the process encoding into the knowledge base, to the process querying, to the automated verification of constraints and to the concern aspectization demonstrate that the automated support provided by the process semantic annotation is useful for business designers' and analysts' activities. Furthermore, the results obtained in the empirical study with human subjects on BPMN VQL demonstrate the actual effectiveness and efficiency (in terms of benefits gained and effort required) of one of the proposed uses of the process semantic annotation. These promising results make us believe that similar benefits could be expected for the other application scenarios (e.g., for the similar BPMN VRL language).

In future studies, we are interested and we plan to empirically investigate the impact of semantic annotation in each application scenario. Moreover, we plan to improve automatic suggestion approaches by exploiting merging axioms and other user-defined constraints. Finally, we are interested in strengthening the support provided to business designers and analysts in the generation of semantically annotated business process models when initial process models do not exist. In detail, we would like to investigate techniques for extending our reverse engineering technique to other (non-Web) types of application interfaces as well as for improving syntax and semantics of element labels in recovered process models, thus also enhancing the quality of the semantic annotation suggestions.

Bibliography

[1] Property-based software engineering measurement. *IEEE Transactions on Software Engineering*, 22(1):68–86, 1996.

[2] Meta-Object Facility (MOF) 1.4 Specification. `http://www.omg.org/technology/documents/formal/mof.htm`, April 2002.

[3] W. Abramowicz, A. Filipowska, M. Kaczmarek, and T. Kaczmarek. Semantically enhanced business process modelling notation. In *Proceedings of the Workshop on Semantic Business Process and Product Lifecycle Management (SBPM), Innsbruck, Austria*, 2007.

[4] M. Adler. An algebra for data flow diagram process decomposition. *Software Engineering, IEEE Transactions on*, 14(2):169 –183, February 1988.

[5] A. V. Aho, R. Sethi, and J. D. Ullman. *Compilers. Principles, Techniques, and Tools.* 1985.

[6] Marco Alberti, Federico Chesani, Marco Gavanelli, Evelina Lamma, Paola Mello, Marco Montali, and Paolo Torroni. Expressing and verifying business contracts with abductive logic programming. *Int. J. Electron. Commerce*, 12:9–38, July 2008.

[7] Mira Mezini Anis Charfi. Aspect-oriented web service composition with AO4BPEL. In *Proceedings of the 2nd European Conference on Web Services (ECOWS)*, volume 3250 of *LNCS*, pages 168–182. Springer, September 2004.

[8] G. Antoniol, G. Canfora, G. Casazza, A. De Lucia, and E. Merlo. Recovering traceability links between code and documentation. *Software Engineering, IEEE Transactions on*, 28(10):970 – 983, October 2002.

[9] Ahmed Awad. Bpmn-q: A language to query business processes. In Manfred Reichert, Stefan Strecker, and Klaus Turowski, editors, *EMISA*, volume P-119 of *LNI*, pages 115–128. GI, 2007.

[10] Ahmed Awad, Gero Decker, and Mathias Weske. Efficient compliance checking using bpmn-q and temporal logic. In *Proceedings of the 6th International Conference on Business Process Management*, BPM '08, pages 326–341, Berlin, Heidelberg, 2008. Springer-Verlag.

[11] Ahmed Awad, Sherif Sakr, and Ghazi Al-Naymat. *Querying Business Processes on Multiple Layers*. IGI Global, 2010.

[12] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[13] Alberto Bacchelli, Marco D'Ambros, Michele Lanza, and Romain Robbes. Benchmarking lightweight techniques to link e-mails and source code. In *Proceedings of the 2009 16th Working Conference on Reverse Engineering*, WCRE '09, pages 205–214, Washington, DC, USA, 2009. IEEE Computer Society.

[14] Amit Basu and Robert W. Blanning. Synthesis and decomposition of processes in organizations. *Info. Sys. Research*, 14:337–355, December 2003.

[15] Andreas Bauer, Martin Leucker, and Christian Schallhart. Monitoring of real-time properties. In S. Arun-Kumar and Naveen Garg, editors, *FSTTCS 2006: Foundations of Software Technology and Theoretical Computer Science*, volume 4337 of *Lecture Notes in Computer Science*, pages 260–272. Springer Berlin / Heidelberg, 2006.

[16] Catriel Beeri, Anat Eyal, Simon Kamenkovich, and Tova Milo. Querying business processes. In *Proceedings of the 32nd International Conference on Very large data bases (VLDB '06)*, pages 343–354, 2006.

[17] Catriel Beeri, Anat Eyal, Simon Kamenkovich, and Tova Milo. Querying business processes with bp-ql. *Information Systems*, 33(6):477–507, 2008.

[18] Ziv Ben-Eliahu and Michael Elhadad. Semantic business process for improved exception handling. Technical report, Department of Computer Science, Ben Gurion University of the Negev, Israel, 2009.

[19] Dominik Birkmeier, Sebastian Kloeckner, and Sven Overhage. An empirical comparison of the usability of bpmn and uml activity diagrams for business users. In *Proceedings of the 18th European Conference on Information Systems (ECIS)*, 2010.

[20] B.W. Boehm, J.R. Brown, and M.L. Lipow. Quantitative evaluation of software quality. In *International Conference on Software Engineering (ICSE)*, 1976.

[21] Andreas Bögl, Michael Schrefl, Gustav Pomberger, and Norbert Weber. Semantic annotation of epc models in engineering domains to facilitate an automated identification of common modelling practices. In Will Aalst, John Mylopoulos, Norman M. Sadeh, Michael J. Shaw, Clemens Szyperski, Joaquim Filipe, and Jos Cordeiro, editors, *Enterprise Information Systems*, volume 19 of *Lecture Notes in Business Information Processing*, pages 155–171. Springer Berlin Heidelberg, 2009.

[22] Matthias Born, Florian Dörr, and Ingo Weber. User-friendly semantic annotation in business process modeling. In Mathias Weske, Mohand-Saïd Hacid, and Claude Godart, editors, *Proceedings of the 2007 International Conference on Web Information Systems Engineering*, volume 4832/2007 of *LNCS*, pages 260–271. Springer-Verlag Berlin, Heidelberg, 2007.

[23] Matthias Born, Jörg Hoffmann, Tomasz Kaczmarek, Marek Kowalkiewicz, Ivan Markovic, James Scicluna, Ingo Weber, and Xuan Zhou. Semantic annotation and composition of business processes with maestro. In *Proceedings of the 5th European semantic web conference on The semantic web: research and applications*, ESWC'08, pages 772–776, Berlin, Heidelberg, 2008. Springer-Verlag.

[24] R.P.J.C. Bose and W.M.P. van der Aalst. Context aware trace clustering: Towards improving process mining results. In *Proc. of Symposium on Discrete Algorithms (SDM-SIAM)*, pages 401–412, USA, 2009.

[25] Business Process Management Initiative (BPMI). Business process modeling notation: Specification. http://www.bpmn.org, 2006.

[26] Mathieu Braem, Kris Verlaenen, Niels Joncheere, Wim Vanderperren, Ragnhild Van Der Straeten, Eddy Truyen, Wouter Joosen, and Viviane Jonckers. Isolating process-level concerns using padus. In Schahram Dustdar, Jos Luiz Fiadeiro, and Amit P. Sheth, editors, *Business Process Management*, volume 4102 of *Lecture Notes in Computer Science*, pages 113–128. Springer, 2006.

[27] G. Canfora, F. García, M. Piattini, F. Ruiz, and C. A. Visaggio. A family of experiments to validate metrics for software process models. *J. Syst. Softw.*, 77:113–129, August 2005.

[28] J. Cardoso, J. Mendling, G. Neumann, and H.A. Reijers. A discourse on complexity of process models. In *Proc. of Workshop on Business Process Intelligence (BPI)*, 2006.

[29] Jorge Cardoso. Process control-flow complexity metric: An empirical validation. *Services Computing, IEEE International Conference on*, 0:167–173, 2006.

[30] Fabio Casati, Silvana Castano, Mariagrazia Fugini, Isabelle Mirbel, and Barbara Pernici. Using patterns to design rules in workflows. *IEEE Trans. Softw. Eng.*, 26:760–785, August 2000.

[31] Tiziana Catarci, Giuseppe Santucci, and Tiziana Catarci. Are visual query languages easier to use than traditional ones? an experimental proof. In *In International Conference on Human-Computer Interaction (HCI95*, pages 323–338, 1995.

[32] Federico Chesani, Paola Mello, Marco Montali, Fabrizio Riguzzi, Maurizio Sebastianis, and Sergio Storari. Checking Compliance of Execution Traces to Business Rules. In Danilo Ardagna, Massimo Mecella, and Jian Yang, editors, *Business Process Management Workshops*, volume 17, chapter 13, pages 134–145. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[33] Fabio Ciravegna, Sam Chapman, Alexiei Dingli, and Yorick Wilks. Learning to harvest information for the semantic web. In *In Proceedings of the 1st European Semantic Web Symposium*, pages 10–12, 2004.

[34] Fabio Ciravegna, Alexiei Dingli, Daniela Petrelli, and Yorick Wilks. User-system cooperation in document annotation based on information extraction. In *In Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management, EKAW02*, pages 122–137. Springer Verlag, 2002.

[35] J. Cohen, editor. *Statistical Power Analysis for the Behavioral Sciences.* Lawrence Erlbaum Assoc., 1988.

[36] S. D. Conte, H. E. Dunsmore, and V. Y. Shen. *Software engineering metrics and models.* Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA, 1986.

[37] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* MIT Press, 1990.

[38] Carine Courbis and Anthony Finkelstein. Towards aspect weaving applications. In *ICSE '05: Proceedings of the 27th international conference on Software engineering*, pages 69–77, New York, NY, USA, 2005. ACM.

[39] Francisco Curbera, Yaron Goland, Yohannes Klein, Frank Leymann, Dieter Roller, and Sanjiva Weerawarana. Business process execution language for web services. Web page. Version 1.0 - July 31, 2002.

[40] O. J. Dahl, E. W. Dijkstra, and C. A. R. Hoare, editors. *Structured programming.* Academic Press Ltd., London, UK, UK, 1972.

[41] A. De Nicola, M. Lezoche, and M. Missikoff. An ontological approach to business process modeling. In *Proceedings of Proceedings of the 3rd Indian International Conference on Artificial Intelligence (IICAI 2007)*, pages 1794–1813, December 2007.

[42] Antonio De Nicola, Michele Missikoff, Maurizio Proietti, and Fabrizio Smith. An open platform for business process modeling and verification. In *Proceedings of the 21st international conference on Database and expert systems applications: Part I*, DEXA'10, pages 76–90, Berlin, Heidelberg, 2010. Springer-Verlag.

[43] Chrysanthos Dellarocas and Mark Klein. A knowledge-based approach for designing robust business processes. In *Business Process Management, Models, Techniques, and Empirical Studies*, pages 50–65. Springer-Verlag, 2000.

[44] Chiara Di Francescomarino, Chiara Ghidini, Marco Rospocher, Luciani Serafini, and Paolo Tonella. Semantically-aided business process modeling. In Abraham Bernstein, David R. Karger, Tom Heath, Lee Feigenbaum, Diana Maynard, Enrico Motta, and Krishnaprasad Thirunarayan, editors, *8th International Semantic Web Conference (ISWC 2009)*, volume 5823/2009 of *Lecture Notes in Computer Science*, pages 114–129, Westfields Conference Center, Washington, DC. USA, 25-29 October 2009. Springer Berlin / Heidelberg.

[45] Chiara Di Francescomarino, Chiara Ghidini, Marco Rospocher, Luciano Serafini, and Paolo Tonella. Reasoning on semantically annotated processes. In *ICSOC*, pages 132–146, 2008.

[46] Chiara Di Francescomarino, Chiara Ghidini, Marco Rospocher, Luciano Serafini, and Paolo Tonella. A framework for the collaborative specification of semantically annotated business processes. *Journal of Software Maintenance and Evolution: Research and Practice*, pages n/a–n/a, 2011.

[47] Chiara Di Francescomarino, Alessandro Marchetto, and Paolo Tonella. Reverse engineering of business processes exposed as web applications. In *13th European*

Conference on Software Maintenance and Reengineering, CSMR 2009, Architecture-Centric Maintenance of Large-SCale Software Systems, Kaiserslautern, Germany, 24-27 March 2009. IEEE, 2009.

[48] Chiara Di Francescomarino, Alessandro Marchetto, and Paolo Tonella. Cluster-based modularization of processes recovered from web applications. *Journal of Software Maintenance and Evolution: Research and Practice*, 2010.

[49] Chiara Di Francescomarino and Paolo Tonella. Crosscutting concern documentation by visual query of business processes. In *Proc. of BPD2008*, 2008.

[50] Chiara Di Francescomarino and Paolo Tonella. Supporting ontology-based semantic annotation of business processes with automated suggestions. In *BM-MDS/EMMSAD*, volume 29 of *Lecture Notes in Business Information Processing*, pages 211–223. Springer, 2009.

[51] Chiara Di Francescomarino and Paolo Tonella. Supporting ontology-based semantic annotation of business processes with automated suggestions. *IJISMD*, 1(2):59–84, 2010.

[52] Giuseppe Antonio Di Lucca, Anna Rita Fasolino, Porfirio Tramontana, and Ugo De Carlini. Recovering a business object model from web applications. In *International Computer Software and Applications Conference (COMPSAC)*. IEEE Computer Society, November 2003.

[53] Remco M. Dijkman, Marlon Dumas, and Chun Ouyang. Formal semantics and automated analysis of bpmn process models, 2007. `http://eprints.qut.edu.au/archive/00005969/`.

[54] M. Dimitrov, A. Simov, S. Stein, and M. Konstantinov. A bpmo based semantic business process modelling environment. In *Proceedings of the Workshop on Semantic Business Process and Product Lifecycle Management (SBPM 2007)*, volume 251 of *CEUR-WS*, 2007.

[55] E.Rolón, L.Sánchez, F.García, F.Ruiz, M.Piattini, D.Caivano, and G.Visaggio. Prediction models for bpmn usability and maintainability. In *Proc. of the International Conference on Commerce and Enterprise Computing (CCEC)*. IEEE Computer Society, 2009.

[56] B. Wetzstein et. al. Semantic business process management: A lifecycle based requirements analysis. In *Proc. of the Workshop on Semantic Business Process and Product Lifecycle Management*, volume 251 of *CEUR Workshop Proceedings*, 2007.

[57] Dieter Fensel. Ontologies: Dynamic networks of formally represented meaning, 2001.

[58] Robert E. Filman, Tzilla Elrad, Siobhn Clarke, and Mehmet Aksit. *Aspect-Oriented Software Development.* Addison-Wesley, October 6, 2004.

[59] Alexander Forster, Gregor Engels, Tim Schattkowsky, and Ragnhild Van Der Straeten. Verification of business process quality constraints based on visual process patterns. In *Proceedings of the First Joint IEEE/IFIP Symposium on Theoretical Aspects of Software Engineering*, pages 197–208, Washington, DC, USA, 2007. IEEE Computer Society.

[60] T. Foss, E. Stensrud, B. Kitchenham, and I. Myrtveit. A simulation study of the model evaluation criterion mmre. *IEEE Transactions on Software Engineering*, 29(11):985–995, 2003.

[61] Martin Fowler. *Refactoring: Improving the Design of Existing Code.* Addison-Wesley, Boston, MA, USA, 1999.

[62] W. B. Frakes and R. Baeza-Yates. *Information Retrieval: Data Structures and Algorithms.* 1992.

[63] A. Gangemi, N. Guarino, C. Masolo, A. Oltramari, and L. Schneider. Sweetening Ontologies with DOLCE. In *Proc. EKAW '02*, volume 2473 of *LNCS*, pages 166–181, 2002.

[64] B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations.* Spring-Verlag, 1999.

[65] C. Ghidini, M. K. Hasan, M. Rospocher, and L. Serafini. A proposal of merging axioms between bpmn and dolce ontologies. Technical report, FBK-irst, 2009. `https://dkm.fbk.eu/index.php/BPMN_Related_Resources`.

[66] C. Ghidini, M. Rospocher, and L. Serafini. A formalisation of BPMN in description logics. Technical Report TR 2008-06-004, FBK-irst, 2008. `https://dkm.fbk.eu/index.php/BPMN_Related_Resources`.

[67] A. Ghose, G. Koliadis, and A. Chueng. Process discovery from model and text artefacts. In *International Workshop on Service- and Process-Oriented Software Engineering (SOPOSE)*. IEEE Computer Society, July 2007.

[68] Stijn Goedertier and Jan Vanthienen. Designing compliant business processes with obligations and permissions. In *Business Process Management Workshops*, pages 5–14, 2006.

[69] Guido Governatori, Zoran Milosevic, and Shazia Sadiq. Compliance checking between business processes and business contracts. In *Proceedings of the 10th IEEE International Enterprise Distributed Object Computing Conference*, pages 221–232, Washington, DC, USA, 2006. IEEE Computer Society.

[70] Guido Governatori and Shazia Sadiq. The journey to business process compliance. In *Handbook of Research on BPM*. IGI Global, 2008.

[71] G. Gröner and S. Staab. Modeling and query patterns for process retrieval in owl. In *Proceedings of the 8th International Semantic Web Conference (ISWC 2009)*, volume 5823 of *LNCS*, pages 243–259. Springer, 2009.

[72] RDF Data Access Working Group. SPARQL query language for RDF. W3C recommendation, W3C, January 2008. `http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/`.

[73] SPARQL Working Group. Sparql new features and rationale. Web page, July 2009. `http://www.w3.org/TR/2010/WD-sparql11-query-20100126/`.

[74] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, June 1993.

[75] Volker Gruhn and Ralf Laue. Complexity metrics for business process models. In *International Conference on Business Information Systems (BIS)*, 2006.

[76] Nicola Guarino. Formal ontology and information systems. pages 3–15. IOS Press, 1998.

[77] Michael Havey. *SOA Cookbook: Design Recipes for Building Better SOA Processes.* October 2008.

[78] Martin Hepp, Frank Leymann, John Domingue, Alexander Wahler, and Dieter Fensel. Semantic business process management: A vision towards using semantic web services for business process management. In *Proc. of ICEBE2005*, 2005.

[79] Heinrich Herre, Barbara Heller, Patryk Burek, Robert Hoehndorf, Frank Loebe, and Hannes Michalek. General Formal Ontology (GFO) – A foundational ontology integrating objects and processes [Version 1.0]. Technical Report 8, Research Group Ontologies in Medicine, Institute of Medical Informatics, Statistics and Epidemiology, University of Leipzig, Leipzig, July 2006.

[80] Matthew Horridge, Bijan Parsia, and Ulrike Sattler. Laconic and precise justifications in owl. In *International Semantic Web Conference*, volume 5318 of *Lecture Notes in Computer Science*, pages 323–338. Springer, 2008.

[81] Maokeng Hung and Ying Zou. Extracting business processes from three-tier architecture systems. In *Reverse Engineering to Requirements (REtR)*. IEEE Computer Society, June 2005.

[82] Maokeng Hung and Ying Zou. Recovering workflows from multi tiered e-commerce systems. In *International Conference on Program Comprehension (ICPC)*. IEEE Computer Society, June 2007.

[83] Nancy Ide and Jean Vronis. Word sense disambiguation: The state of the art. *Computational Linguistics*, 24:1–40, 1998.

[84] Business Process Management Initiative. Business process: Business process query language. `http://www.service-architecture.com/web-services/articles/business_process_query_language_bpql.html`.

[85] Doug Janzen and Kris De Volder. Navigating and querying code without getting lost. In *AOSD '03: Proceedings of the 2nd international conference on Aspect-oriented software development*, pages 178–187, New York, NY, USA, 2003. ACM Press.

[86] M. Jorgensen. Experience with the accuracy of software maintenance task effort prediction models. *IEEE Transactions on Software Engineering*, 21(8):674–681, 1995.

[87] S.K. Kachingan, editor. *Statistical Analysis*. Radius Press, 1986.

[88] Stephen H. Kan. *Metrics and Models in Software Quality Engineering*. Addison-Wesley, 2003.

[89] G. Keller, M. Nüttgens, and A. W. Scheer. Semantische Prozeßmodellierung auf der Grundlage Ereignisgesteuerter Prozeßketten (EPK). Technical Report 89, Universität des Saarlandes, Germany, Saarbrücken, Germany, January 1992.

[90] K.-H. Kim and Y.-G. Kim. Process reverse engineering for bpr: A form-based approach. *Journal of Information and Management*, 23(4):187 – 200, 1998.

[91] Paul Kogut and William Holmes. Aerodaml: Applying information extraction to generate daml annotations from web pages. In *First International Conference on Knowledge Capture (K-CAP 2001). Workshop on Knowledge Markup and Semantic Annotation*, 2001.

[92] Woralak Kongdenfha, Rgis Saint-Paul, Boualem Benatallah, and Fabio Casati. An aspect-oriented framework for service adaptation. In Asit Dan and Winfried Lamersdorf, editors, *Service-Oriented Computing ICSOC 2006*, volume 4294 of *Lecture Notes in Computer Science*, pages 15–26. Springer Berlin / Heidelberg, 2006.

[93] A. Koschmider and A. Oberweis. Ontology based business process description. In *Proceedings of the CAiSE-05 Workshops*, LNCS, pages 321–333. Springer, 2005.

[94] John Krogstie. Eeml2005: Extended enterprise modeling language. Technical report, Norwegian University of Science and Technology, Norway, 2005.

[95] Markus Krotzsch, Denny Vrandecic, and Max Volkel. Wikipedia and the semantic web - the missing links. In *Proceedings of the 1st Internation Wikimedia Conference (Wikimania 2005)*, 2005.

[96] Dominj Kuropka. Modelle zur reprsentation natrlichsprachlicher dokumente: Information-filtering und -retrieval mit relationalen datenbanken (in german). Information Systems and Management Science, 10th issue. Logos Verlag, Berlin, 2004.

[97] Ramnivas Laddad. *AspectJ in Action: Practical Aspect-Oriented Programming.* Manning, July 2003.

[98] F. Lautenbacher, B. Bauer, and C Seitz. Semantic business process modeling - benefits and capability. In *AAAI 2008 Stanford Spring Symposium - AI Meets Business Rules and Process Management (AIBR)*, 2008.

[99] Douglas B. Lenat and R. V. Guha. The evolution of cycl, the cyc representation language. *SIGART Bull.*, 2:84–87, June 1991.

[100] Barbara Staudt Lerner, Stefan Christov, Leon J. Osterweil, Reda Bendraou, Udo Kannengiesser, and Alexander Wise. Exception handling patterns for process modeling. *IEEE Transactions on Software Engineering*, 99(RapidPosts):162–183, 2010.

[101] Dekang Lin. An information-theoretic definition of similarity. In *ICML'98*, pages 296–304, 1998.

[102] Yun Lin. *Semantic Annotation for Process Models: Facilitating Process Knowledge Management via Semantic Interoperability.* PhD thesis, Norwegian University of Science and Technology, Trondheim, Norway, 2008.

[103] Y. Liu, S. Müller, and K. Xu. A static compliance-checking framework for business process models. *IBM Syst. J.*, 46:335–361, April 2007.

[104] John W. Lloyd. *Foundations of logic programming.* Springer-Verlag New York, Inc., 1987.

[105] Thomas W. Malone, Kevin Crowston, and George A. Herman. *Organizing Business Knowledge: The MIT Process Handbook.* MIT Press, Cambridge, MA, USA, 2003.

[106] S. Mancoridis, B. S. Mitchell, Y. Chen, and E. R. Gansner. Bunch: a clustering tool for the recovery and maintenance of software system structures. pages 50–59, Oxford, England, 1999.

[107] Chrisotpher D. Manning and Hinrich Schtze. *Foundations of Statistical Natural Language Processing.* The Mit Press, Cambridge MA, 1999.

[108] Alessandro Marchetto, Roberto Tiella, Paolo Tonella, Nadia Alshahawan, and Mark Harman. Crawlability metrics for automated web testing. *(to appear) Journal of Software Tools for Technology Transfer*, 2010.

[109] Ivan Markovic. Advanced querying and reasoning on business process models. In *Proceedings of the 11th International Conference on Business Information Systems (BIS 2008*, LNBIP, pages 189–200. Springer, 2008.

[110] Joachim Melcher, Jan Mendling, Hajo A. Reijers, and Detlef Seese. On measuring the understandability of process models. In *Proc. of Workshop on Empirical Research in Business Process Management (ER-BPM)*, 2009.

[111] J. Mendling, H. Reijers, and J. Cardoso. What makes process models understandable? In *Business Process Management (BPM)*, pages 117–128. Springer, 2007.

[112] J. Mendling, H. A. Reijers, and W. M. P. van der Aalst. Seven process modeling guidelines (7pmg). *Inf. Softw. Technol.*, 52:127–136, February 2010.

[113] J. Mendling and H.A. Reijers. How to define activity labels for business process models? In *SIGSAND Europe2008*, pages 117–128, 2008.

[114] Jan Mendling. Detection and prediction of errors in epc business process models. *Ph.D. Thesis on Vienna University of Economics and Business Administration*, 2007.

[115] Jan Mendling and Jan Recker. Towards systematic usage of labels and icons in business process models. In *Proc. of EMMSAD2008*, volume 337 of *CEUR WS*, pages 1–13. T. Halpin, E. Proper, J. Krogstie, X. Franch, E. Hunt, R. Coletta, eds., June 2008.

[116] Jan Mendling and Hajo A. Reijers. The impact of activity labeling styles on process model quality. In *SIGSAND-EUROPE*, pages 117–128, 2008.

[117] Lawrence D. Miles. *Techniques of Value Analysis and Engineering.* Mcgraw-Hill (Tx), 1972.

[118] George A. Miller. Wordnet: a lexical database for english. *Commun. ACM*, 38(11):39–41, 1995.

[119] Michele Missikoff, Maurizio Proietti, and Fabrizio Smith. A Business Process Knowledge Base for Composite Services Development. In *Proceedings of International Workshop on Business System Management and Engineering (BSME 2010)*, 2010.

[120] Mariusz Momotko and Kazimierz Subieta. Process Query Language: A Way to Make Workflow Processes More Flexible. In András Benczúr, János Demetrovics, and Georg Gottlob, editors, *Proceeding of the 8th East European Conference on Advances in Databases and Information Systems (ADBIS 2004)*, pages 306–321, September 2004.

[121] Marco Montali. *Specification and Verification of Declarative Open Interaction Models: a Logic-Based Approach*, volume 56 of *Lecture Notes in Business Information Processing.* Springer, 2010.

[122] M. Monteiro and J. Fernandes. Towards a catalog of aspect-oriented refactorings. In Mira Mezini and Peri L. Tarr, editors, *Proceedings of the 4th International Conference on Aspect-Oriented Software Development, AOSD 2005, Chicago, Illinois, USA, March 14-18, 2005*, pages 111–122. ACM, 2005.

[123] D. Moody. Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions. *Data and Knowledge Engineering*, 55:243–276, 2005.

[124] H.R. Motahari-Nezhad, R. Saint-Paul, B. Benatallah, and F. Casati. Deriving protocol models from imperfect service conversation logs. *Knowledge and Data Engineering, IEEE Transactions on*, 20(12):1683 –1698, 2008.

[125] Michael Zur Muehlen and Jan Recker. How much language is enough? theoretical and practical use of the business process modeling notation. In *Proceedings of the 20th international conference on Advanced Information Systems Engineering*, CAiSE '08, pages 465–479, Berlin, Heidelberg, 2008. Springer-Verlag.

[126] G. C. Murphy, D. Notkin, and K. Sullivan. Software reflexion models: Bridging the gap between source and high-level models. In *Proceedings of the Third ACM Symposium on the Foundations of Software Engineering*, pages 18–28, 1995.

[127] I. Myrtveit, E. Stensrud, and M. Shepperd. Reliability and validity in comparative studies of software prediction models. *IEEE Transactions on Software Engineering*, 31(5):380–39, 2005.

[128] Kioumars Namiri and Nenad Stojanovic. Pattern-Based Design and Validation of Business Process Compliance. In Robert Meersman and Zahir Tari, editors, *On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS*, volume 4803 of *Lecture Notes in Computer Science*, chapter 6, pages 59–76. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[129] Jörg Nitzsche, Daniel Wutke, and Tammo van Lessen. An ontology for executable business processes. In *Proceedings of the Workshop on Semantic Business Process and Product Lifecycle Management held in conjunction with the 3rd European Semantic Web Conference (ESWC 2007), Innsbruck, Austria, June 7, 2007*, 2007.

[130] Tommaso Di Noia, Eugenio Di Sciascio, Francesco M. Donini, Eufemia Tinelli, Francesco di Cugno, and Azzurra Ragone. Automated building blocks selection based on business processes semantics in erps. *Service Oriented Computing and Applications*, 1(3):171–184, 2007.

[131] OMG. Business process modeling notation, v1.1. http://www.bpmn.org.

[132] OMG. Unified modelling language, v2.3. `http://www.omg.org/spec/UML/2.3/`.

[133] OMG. Owl 2: Web ontology language. `http://www.w3.org/TR/owl2-overview/`, 2004.

[134] OMG. Owl: Web ontology language. `http://www.w3.org/TR/owl-ref/`, 2004.

[135] OMG. Business process modeling notation (bpmn) version 1.2. http://www.bpmn.org, 2009.

[136] OMG. Business process modeling notation (bpmn) version 2.0. http://www.bpmn.org, 2011.

[137] C. Ouyang, W.M.P. van der Aalst, M. Dumas, and A.H.M. ter Hofstede. Traslating bpmn to bpel. Technical report, Queensland University of Technology, The Netherlands and Eindhoven University of Technology, Australia, 2006.

[138] Bronius Paradauskas and Aurimas Laurikaitis. Business knowledge extraction from legacy information systems. *Journal of Information Technology and Control*, 35(3):214 – 221, 2006.

[139] Abhijit Patil, Swapna Oundhakar, Amit Sheth, and Kunal Verma. Meteor-s web service annotation framework. In *In Proceedings of the 13th International Conference on the World Wide Web*, pages 553–562. ACM Press, 2004.

[140] M. Pesic and W. van der Aalst. A Declarative Approach for Flexible Business Processes Management. pages 169–180. 2006.

[141] James L. Peterson. *Petri Net Theory and the Modeling of Systems.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 1981.

[142] Borislav Popov, Atanas Kiryakov, Damyan Ognyanoff, Dimitar Manov, and Angel Kirilov. Kim a semantic platform for information extraction and retrieval. *Nat. Lang. Eng.*, 10:375–392, September 2004.

[143] J. Recker, M. Indulska, M. Rosemann, and P. Green. How Good is BPMN Really? Insights from Theory and Practice. pages 1582–1593. Association for Information Systems, 2006.

[144] Jan Recker. Opportunities and constraints: the current struggle with BPMN. *Business Process Management Journal*, 16(1):181–201, 2010.

[145] Jan Recker, Michael Rosemann, Marta Indulska, and Peter F. Green. Business process modeling- a comparative analysis. *J. AIS*, 10(4), 2009.

[146] Jan C. Recker and Alexander Dreiling. Does it matter which process modelling language we teach or use? An experimental study on understanding process modelling languages without formal education, 2007.

[147] Jan C. Recker, Michael Zur Muehlen, Siau Keng, J. Erickson, and M. Indulska. Measuring Method Complexity: UML versus BPMN. 2009.

[148] H. A. Reijers and J. Mendling. A study into the factors that influence the understandability of business process models. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, PP(99):1 –14, 2010.

[149] Hajo Reijers and Jan Mendling. Modularity in process models: Review and effects. In *Business Process Management (BPM)*, 2008.

[150] Hajo Reijers, Jan Mendling, and Remco Dijkman. On the usefulness of subprocesses in business process models. Technical report, 2010.

[151] Edwina L. Rissland, David B. Skalak, and M. Timur Friedman. Evaluating a legal argument program: The bankxx experiments. *Artif. Intell. Law*, 5(1-2):1–74, 1997.

[152] Martin P. Robillard and Gail C. Murphy. Concern graphs: finding and describing concerns using structural program dependencies. In *ICSE '02: Proceedings of the 24th International Conference on Software Engineering*, pages 406–416, New York, NY, USA, 2002. ACM.

[153] Marco Rospocher, Chiara Di Francescomarino, Chiara Ghidini, Luciano Serafini, and Paolo Tonella. Collaborative specification of semantically annotated business processes. In Stefanie Rinderle-Ma, Shazia Sadiq, and Frank Leymann, editors, *Business Process Management Workshops - BPM 2009 International Workshops*, volume 43 of *Lecture Notes in Business Information Processing*, pages 305–317. Springer, 2010.

[154] A. Rozinat, A.K. Alves de Medeiros, C.W. Gunther, A.J.M.M. Weijters, and W.M.P. van der Aalst. Towards an evaluation framework for process mining algorithms. Technical report, Technical report Eindhoven University of Technology, The Netherlands., 2007.

[155] A. Rozinat and W.M.P. van der Aalst. Conformance checking of processes based on monitoring real behavior. *Information Systems*, 33(1):64–95, 2008.

[156] Anne Rozinat and Wil M. P. van der Aalst. Conformance testing: Measuring the fit and appropriateness of event logs and process models. In *Business Process Management Workshops*, pages 163–176, 2005.

[157] Nick Russell, Wil M. P. van der Aalst, and Arthur H. M. ter Hofstede. Exception handling patterns in process-aware information systems. Technical report, BPM-center.org, 2006.

[158] Arun Anand Sadanandan, Kow Weng Onn, and Dickson Lukose. Ontology based graphical query language supporting recursion. In *Proceedings of the 14th international conference on Knowledge-based and intelligent information and engineering systems: Part I*, KES'10, pages 627–638, Berlin, Heidelberg, 2010. Springer-Verlag.

[159] Rainer Schmidt, Christian Bartsch, and Roy Oberhauser. Ontology-based representation of compliance requirements for service processes. In *SBPM*, 2007.

[160] Alec Sharp and Patrick Mcdermott. *Workflow Modeling: Tools for Process Improvement and Application Development.* Artech House Publishers, 2001.

[161] Evren Sirin and Jiao Tao. Towards integrity constraints in owl. In *OWLED*, volume 529 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.

[162] Howard Smith and Peter Fingar. *Business Process Management: The Third Wave.* Meghan-Kiffer Press, 2003.

[163] Veda C. Storey. Comparing relationships in conceptual modeling: Mapping to semantic classifications. *IEEE Trans. on Knowl. and Data Eng.*, 17(11):1478–1489, 2005.

[164] Kazimierz Subieta, Catriel Beeri, Florian Matthes, and Joachim W. Schmidt. A stack-based approach to query languages. In *East/West Database Workshop'94*, pages 159–180, 1994.

[165] Stanley Sutton Jr. and Isabelle Rouvellou. Modeling of software concerns in Cosmos. In Gregor Kiczales, editor, *Proc. 1st Int' Conf. on Aspect-Oriented Software Development (AOSD-2002)*, pages 127–133. ACM Press, April 2002.

[166] Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146 – 160, 1972.

[167] O. Thomas and M. Fellmann. Semantic epc: Enhancing process modeling using ontology languages. In *Proceedings of the Workshop on Semantic Business Process and Product Lifecycle Management (SBPM 2007)*, pages 64–75, June 2007.

[168] T.L.Alves, C.Ypma, and J.Visser. Deriving metric thresholds from benckmark data. In *Proc. of the International Conference on Software Maintenance (ICSM)*. IEEE Computer Society, 2010.

[169] Paolo Tonella and Giuliano Antoniol. Object-oriented design pattern inference. In *ICSM*, pages 230–, 1999.

[170] Paolo Tonella and Mariano Ceccato. Aspect mining through the formal concept analysis of execution traces. In *WCRE '04: Proceedings of the 11th Working Conference on Reverse Engineering (WCRE'04)*, pages 112–121, Washington, DC, USA, 2004. IEEE Computer Society.

[171] Tom Tourw and Kim Mens. Mining aspectual views using formal concept analysis. In *Proceedings of the 4th IEEE International Workshop on Source Code Analysis and Manipulation (SCAM 2004)*. IEEE Computer Society, 2004.

[172] Victoria Uren, Philipp Cimiano, Jose Iria, Siegfried Handschuh, Maria Vargas-Vera, Enrico Motta, and Fabio Ciravegna. Semantic annotation for knowledge management: Requirements and a survey of the state of the art. *Web Semantics: Science, Services and Agents on the World Wide Web*, 4(1):14–28, January 2006.

[173] W. M. P. van der Aalst, V. Rubin, H. M. W. Verbeek, B.F. van Dongen, E. Kindler, and C. W. Gnther. Process mining: a two-step approach to balance between underfitting and overfitting. *Journal of Software and Systems Modeling*, 9(1):87–111, 2008.

[174] W. M. P. van der Aalst and A. H. M. ter Hofstede. Yawl: yet another workflow language. *Inf. Syst.*, 30:245–275, June 2005.

[175] Wil M. P. van der Aalst, Arthur, and Mathias Weske. Business Process Management: A Survey. *Lecture Notes in Computer Science*, 2678:1–12, January 2003.

[176] Wil M. P. van der Aalst, H. T. de Beer, and Boudewijn F. van Dongen. Process mining and verification of properties: An approach based on temporal logic. In *OTM Conferences (1)*, pages 130–147, 2005.

[177] Wil M. P. Van Der Aalst, Hajo A. Reijers, and Minseok Song. Discovering social networks from event logs. *Comput. Supported Coop. Work*, 14(6):549–593, December 2005.

[178] W.M.P. van der Aalst, B.F. van Dongen, J. Herbst, L. Maruster1, G.Schimm, and A.J.M.M. Weijters. Workflow mining: A survey of issues and approaches. *Journal of Data and Knowledge Engineering*, 47(2):237 – 267, 2003.

[179] W.M.P. van der Aalst, A.J.M.M. Weijter, and L. Maruster. Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, 16:2004, 2003.

[180] Irene Vanderfeesten, Jorge Cardoso, Jan Mendling, Hajo A. Reijers, and Wil van der Aalst. Quality metrics for business process models. In *Proc. of BPM and Workflow Handbook*, pages 179–190, 2007.

[181] Irene Vanderfeesten, Hajo A. Reijers, Jan Mendling, Wil M. P. van der Aalst, and Jorge Cardoso. On a quest for good process models: The cross-connectivity metric. In *Conference on Advanced Information Systems Engineering (CAISE)*, 2008.

[182] Jussi Vanhatalo, Hagen Vlzer, and Jana Koehler. The refined process structure tree. In *International Conference on Business Process Management (BPM)*, 2008.

[183] Jussi Vanhatalo, Hagen Völzer, and Frank Leymann. Faster and more focused control-flow analysis for business process models through sese decomposition. In *Proceedings of the 5th international conference on Service-Oriented Computing*, ICSOC '07, pages 43–55, Berlin, Heidelberg, 2007. Springer-Verlag.

[184] Maria Vargas-vera, Enrico Motta, John Domingue, Mattia Lanzoni, Arthur Stutt, and Fabio Ciravegna. Mnm: A tool for automatic support on semantic markup, kmi. Technical report, 2003.

[185] Gabriel M. Veiga and Diogo R. Ferreira. Understanding spaghetti models with sequence clustering for prom. In *Proc. of Workshop on Business Process Intelligence (BPI)*, 2009.

[186] Bart Verheecke, Mara Agustina Cibrán, and Viviane Jonckers. Aspect-oriented programming for dynamic web service monitoring and selection. In Liang-Jie Zhang, editor, *ECOWS*, volume 3250 of *Lecture Notes in Computer Science*, pages 15–29. Springer, 2004.

[187] Xia Wang. owsd: A tool for word sense disambiguation in its ontology context. In *International Semantic Web Conference (Posters & Demos)*, 2008.

[188] Xiaodong Wang, Nan Li, Hongming Cai, and Boyi Xu. An ontological approach for semantic annotation of supply chain process models. In Robert Meersman, Tharam Dillon, and Pilar Herrero, editors, *On the Move to Meaningful Internet Systems: OTM 2010*, volume 6426 of *Lecture Notes in Computer Science*, pages 540–554. Springer Berlin / Heidelberg, 2010.

[189] Barbara Weber, Manfred Reichert, and Stefanie Rinderle-Ma. Change patterns and change support features - enhancing flexibility in process-aware information systems. *Data Knowl. Eng.*, 66:438–466, September 2008.

[190] Ingo Weber, Joerg Hoffmann, and Jan Mendling. Semantic business process validation. In *Proceedings of the Workshop on Semantic Business Process and Product Lifecycle Management (SBPM 2008)*, June 2008.

[191] Mathias Weske. *Business Process Management: Concepts, Languages, Architectures.* Springer, 2007.

[192] Sthephen A. White and Derek Miers. *BPMN Modeling and Reference Guide. Understanding and Using BPMN.* Future Strategies Inc., Lighthouse Pt, FL, 2008.

[193] Hugh Williams and David Lane, editors. *Web Database Applications with PHP, and MySQL.* O'Reilly Media, 2002.

[194] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Bjöorn Regnell, and Anders Wesslén. *Experimentation in software engineering: an introduction.* Kluwer Academic Publishers, Norwell, MA, USA, 2000.

[195] P. Wong and J. Gibbons. A Relative Timed Semantics for BPMN. Submitted. Extended version available at `http://web.comlab.ox.ac.uk/oucl/work/peter.wong/pub/bpmntime.pdf`, 2008.

[196] Y. Zhou and H. Leung. Predicting object-oriented software maintainability using multivariate adaptive regression splines. *Journal of Systems and Software*, 80(8):1349–1361, 2007.

[197] Ying Zou, Jin Guo, King Chun Foo, and Maokeng Hung. Recovering business processes from business applications. *Journal of Software Maintenance and Evolution: Research and Practice*, 21(5):315–348, 2009.

[198] Ying Zou, Terence C. Lau, Kostas Kontogiannis, Tack Tong, and Ross McKegney. Model driven business process recovery. In *Working Conference on Reverse Engineering (WCRE).* IEEE Computer Society, June 2004.

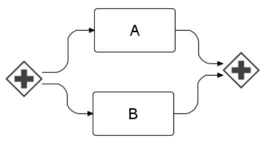
[199] Ying Zou, Qi Zhang, and Xulin Zhao. Improving the usability of e-commerce applications using business processes. *IEEE Transactions on Software Engineering*, 33(12):837 – 855, 2007.

Appendix A

Empirical Study Pre-questionnaire

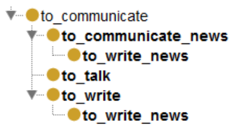
Pre-Questionnaire.

1. How are the two tasks (A and B) in the following fragment of BPMN process executed?



(a) A is always executed before B;
(b) A is always executed after B;
(c) Either A or B is executed;
(d) A and B are executed in parallel;
(e) A or B or both can be executed.

2. By considering the following fragment of ontology, which are all the superconcepts (according to the is_a relationship) of the concept "to_write_news"?



3. How long have you been modeling processes in BPMN or in any other process modeling language?
(a) less than 6 months;
(b) between 6 months and 1 year;
(c) between 1 and 3 years;
(d) between 3 and 5 years;
(e) more than 5 years.

4. How long have you been modeling (or using) ontologies?
(a) less than 6 months;
(b) between 6 months and 1 year;
(c) between 1 and 3 years;
(d) between 3 and 5 years;
(e) more than 5 years.

5. How long have you been using visual languages such as UML, Tropos, BPEL (e.g., for designing, programming, …)?
(a) less than 6 months;
(b) between 6 months and 1 year;
(c) between 1 and 3 years;
(d) between 3 and 5 years;
(e) more than 5 years.

Data collected will be used only for research purposes and they will be revealed only in aggregated form.

Appendix B

Empirical Study Post-questionnaire

Natural Language Final Questionnaire.

1. Understanding the query description in natural language has been:
   (a) immediate; (b) easy; (c) reasonable; (d) difficult; (e) complex.
2. Understanding ontology and ontology concepts, when used, in natural language query has been:
   (a) immediate; (b) easy; (c) reasonable; (d) difficult; (e) complex.
3. Matching the query in the process has been:
   (a) immediate; (b) easy; (c) reasonable; (d) difficult; (e) complex.
4. What are the main difficulties you found in understanding natural language queries?

BPMN VQL Final Questionnaire.

1. Understanding BPMN VQL queries has been:
   (a) immediate; (b) easy; (c) reasonable; (d) difficult; (e) complex.
2. Understanding ontology and ontology concepts, when used, in BPMN VQL queries has been:
   (a) immediate; (b) easy; (c) reasonable; (d) difficult; (e) complex.
3. Matching the query in the process has been:
   (a) immediate; (b) easy; (c) reasonable; (d) difficult; (e) complex.
4. The BPMN-VQL language training has been:
   (a) very useful; (b) useful; (c) not relevant; (d) not useful; (e) counterproductive.
5. How do you judge your understanding of the BPMN VQL?
   (a) very low; (b) low; (c) medium; (d) high; (e) very high.
6. How do you judge the effort required for understanding queries in BPMN VQL?
   (a) very low; (b) low; (c) medium; (d) high; (e) very high.
7. What are the main difficulties you found in understanding BPMN-VQL queries?

8. Understanding query formulation specifications has been:
   (a) immediate; (b) easy; (c) reasonable; (d) difficult; (e) complex.
9. Using the BPMN VQL for formulating the query has been:
   (a) immediate; (b) easy; (c) reasonable; (d) difficult; (e) complex.
10. How do you judge the effort required in writing queries in BPMN VQL?
    (a) very low; (b) low; (c) medium; (d) high; (e) very high.
11. How do you judge the BPMN-VQL expressive power?
    (a) very low; (b) low; (c) medium; (d) high; (e) very high.
12. Querying support to designers and analysts in understanding and maintaining processes is:
    (a) very low; (b) low; (c) medium; (d) high; (e) very high.
13. What are the main difficulties you found in writing queries in BPMN VQL?

14. How could we improve BPMN-VQL?

Overall Question.

Do you think that the benefits of BPMN VQL justify the effort involved in formulating the BPMN VQL queries?
(a) not at all; (b) for a small part; (c) partially; (d) mostly; (e) definitively.