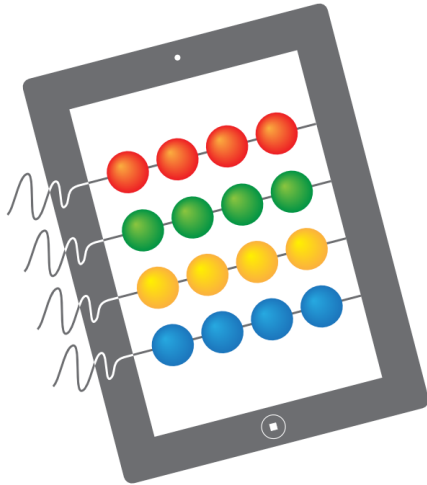




FP7 ICT STREP Project



LEARN PAD

Deliverable D 4.1

Formal Verification of Business Processes

Formal Verification of Business Processes

<http://www.learnpad.eu>



LINAGORA



No Magic Europe

n|w

Fachhochschule
Nordwestschweiz



UNIVERSITY OF GENEVA
1794

*WIKI



Project Number	: FP7-619583
Project Title	: Learn PAd Model-Based Social Learning for Public Administrations

Deliverable Number	: D 4.1
Title of Deliverable	: Formal Verification of Business Processes
Nature of Deliverable	: Other
Dissemination level	: Public
Licence	: Creative Commons Attribution 3.0 License
Version	: 4.0
Contractual Delivery Date	: July 24, 2015
Actual Delivery Date	: August 4, 2015
Contributing WP	: WP4
Editor(s)	: Barbara Re (UniCam), Fabrizio Fornari (UniCam), Giorgio Oronzo Spagnolo (CNR)
Author(s)	: Barbara Re (UniCam), Andrea Polini (UniCam), Stefania Gnesi (CNR), Alessio Ferrari (CNR), Fabrizio Fornari (UniCam), Giorgio O. Spagnolo (CNR), Flavio Corradini (UniCam)
Reviewer(s)	: Guglielmo De Angelis (CNR), Jean Simard (xWIKI)

Abstract

In this deliverable we identified the quality assessment strategies for Business Process models related to the Learn PAd project. This is discussed considering different roles impacting on the platform and strategies to be implemented for its maintaining. After an in-depth analysis of the literature this deliverable (i) reports guidelines to model Business Processes in such a way that resulting model are understandable and (ii) discusses Business Process correctness criteria. Such quality check impacts on the learnability of the Business Process for Public Administrations. The deliverable also introduces some technical details that paves the basis to successively create the Learn PAd platform model verification component.

Keyword List

BPMN, Business Process, Understandability, Verification, Quality, Learnability

Document History

Version	Changes	Author(s)
1.0	First Draft - Table of Content.	Stefania Gnesi, Giorgio O. Spagnolo, Barbara Re
1.2	First draft state of the art on process understandability.	Giorgio O. Spagnolo
1.3	First draft understandability guideline.	Giorgio O. Spagnolo
1.4	First draft state of the art on process correctness.	Fabrizio Fornari
1.5	First draft business process correctness.	Fabrizio Fornari
1.6	First draft quality assessment strategy for BP model.	Alessio Ferrari
2.0	First Draft Introduction.	Barbara Re
2.1	First draft architectural view on verification component	Andrea Polini
3.0	Detailed description of the state of the art has been introduced as well as a complete description of understandability guidelines and business process correctness.	Barbara Re, Andrea Polini, Stefania Gnesi, Alessio Ferrari, Fabrizio Fornari, Giorgio O. Spagnolo, Flavio Corradini
4.0	Reviewer comments have been addressed.	Stefania Gnesi, Alessio Ferrari, Fabrizio Fornari, Giorgio O. Spagnolo, Barbara Re

Document Reviews

Release	Date	Ver.	Reviewers	Comments
ToC	1 June 2015	1.0		
Draft	26 June 2015	2.0		
Internal	10 July 2015	3.0	Guglielmo De Angelis (CNR), Jean Simard (xWIKI)	Details are available in the individual review report.
Candidate Final	24 July 2015	4.0		

Glossary, acronyms & abbreviations

Item	Description
7PMG	Seven Process Modeling Guidelines
AGD	Average Gateway Degree
BP	Business Process
BPEL	Business Process Execution Language
BPMN	Business Process Model and Notation
CFC	Control Flow Complexity
GM	Gateway Mismatch
GH	Gateway Heterogeneity
CRG	Compliance Rule Graph
eCRG	extended Compliance Rule Graph
CTL	Computation Tree Logic
G-CTL	Graphical Computational Tree Logic
EG-CTL	Extended Graphical Computational Tree Logic
EPC	Event Process Chain
FCL	Formal COntract Language
FSP	Finite State Processes
LTL	Linear Temporal Logic
MGD	Maximum Gateway Degree
NL	Natural Language
OMG	Object Management Group
PA	Public Administration
PAIS	Process-Aware Information System
PN	Petri Net
TNG	Total Number of Gateways

Table Of Contents

List Of Tables	XI
List Of Figures	XV
1 Introduction	1
1.1 <i>Deliverable Purpose</i>	1
1.2 <i>Related Deliverables</i>	2
1.3 <i>Deliverable Structure</i>	2
2 Quality Assessment Strategy for Business Process Models	3
2.1 <i>Roles and Tasks</i>	3
2.2 <i>Quality Evaluation Process</i>	5
2.2.1 <i>Automated Quality Assessment Strategy</i>	6
2.2.2 <i>Crowd-based Quality Assessment Strategy</i>	6
3 Overview and State of the Art	11
3.1 <i>Process Model Understandability</i>	11
3.1.1 <i>Business Process Modeling Guidelines</i>	11
3.1.2 <i>Business Process Model Metrics</i>	13
3.1.3 <i>Threshold for Business Process Model Metrics</i>	13
3.2 <i>Business Process Formal Verification</i>	17
3.2.1 <i>Petri Net Based Approaches</i>	17
3.2.2 <i>Model Checker Input Language</i>	17
3.2.3 <i>Business Process Compliance</i>	18
4 Models Understandability	21
4.1 <i>Understandability Guidelines</i>	21
4.1.1 <i>Categories</i>	21
4.1.2 <i>General Guidelines</i>	23
4.1.3 <i>Notation Usage Guidelines</i>	29
4.1.4 <i>Labeling Guidelines</i>	45
4.1.5 <i>Patterns Guidelines</i>	53
4.1.6 <i>Appearance Guidelines</i>	56
4.2 <i>Understandability into Practice</i>	64
5 Business Process Correctness	69
5.1 <i>Business Process Formalization</i>	69
5.2 <i>Correctness Verification</i>	73
5.2.1 <i>Control Flow Correctness</i>	73
5.2.2 <i>Unfolding</i>	74

5.3	<i>Further Developments</i>	78
5.3.1	<i>Data Flow Correctness</i>	78
5.3.2	<i>Compliance</i>	79
6	Architectural View on Verification Component	87
6.1	<i>Verification Component Architecture Overview</i>	87
6.1.1	<i>Business Process to Petri Net Generator</i>	88
6.1.2	<i>Optimization Engine</i>	88
6.1.3	<i>Model Checker Component</i>	88
6.1.4	<i>Metrics Calculator</i>	89
6.2	<i>Interaction Flows</i>	89
6.2.1	<i>External Flows</i>	89
6.2.2	<i>Internal Flows</i>	90
7	Conclusions and Future Works	95
A	Appendix Business Process Model Metrics	97
B	Appendix Business Process Verification Approaches	103
C	Appendix Background on Petri Net and Occurrence Net	107
D	Appendix extended Compliance Rule Graphs	109
	Bibliography	111

List Of Tables

Table 3.1: Seven process modeling guidelines (Mendling et. al. [54]).....	12
Table 3.2: Ten process modeling rules (Mendling et. al. [55].	12
Table 3.3: Understandability and Modifiability Measures.	14
Table 3.4: Threshold values for conceptual model metrics (Snchez et. al. [75])	15
Table 3.5: Thresholds identified based on ROC Curves (Mendling et. al. [55])	15
Table 3.6: Threshold values and linguistic labels for gateway complexity measures (Sanchez et. al. [76]).....	16
Table 4.1: Template of description guidelines of Business Process Model Notation.	22
Table 5.1: Unfolding of the basic example.	77
Table 5.2: Transitions in conflict with the cut-off.....	78
Table A.1: Business Process Model Complexity Metrics. Part 1.	99
Table A.2: Business Process Model Complexity Metrics. Part 2.	100
Table A.3: Business Process Model Complexity Metrics. Part 3.	101
Table A.4: Business Process Model Complexity Metrics. Part 4.	102
Table B.1: List of sources from the field of Business Process Verification. Part 1.	105
Table B.2: List of sources from the field of Business Process Verification. Part 2.	106

List Of Figures

Figure 2.1: Roles and Tasks associated to the quality assessment strategy for BP Models.....	3
Figure 2.2: The two quality assessment strategies for BP Models depicted as components of an overall quality assessment process.	5
Figure 2.3: Example of crowd-based quality assessment of the models. The Learner highlights a missing document from the model, and the Content Manager provides him an early feedback. Then, he will contact the Modeller to update the model.....	7
Figure 2.4: Example of crowd-based quality assessment of the models. The Modeler updates the model according to the request of the Content Manager.....	8
Figure 2.5: Example of crowd-based quality assessment of the models. The Modeler re-updates the model, to increase the readability.....	9
Figure 4.1: Monti Azzurri Consortium SCIA Commerciale BP of the Municipality Fig. 4.25 of WP8.1.....	64
Figure 4.2: Monti Azzurri Consortium SCIA Commerciale BP of the Municipality.....	65
Figure 4.3: Monti Azzurri Consortium “SCIA Commerciale” BP of the Municipality.....	66
Figure 4.4: Monti Azzurri Consortium “Variante Urbanistica” BP Fig. 4.39 of WP8.1.....	67
Figure 4.5: Monti Azzurri Consortium “Variante Urbanistica” BP of the Municipality.	68
Figure 5.1: Selected mapping from BPMN to Petri Net (Dijkman et al. [18]).	70
Figure 5.2: Mapping of a sub-process without exception handling (Dijkman et al. [18]).	70
Figure 5.3: Mapping of message flows between BPMN processes. (Dijkman et al. [18]).	71
Figure 5.4: Mapping Participant B: start event and gateway.	71
Figure 5.5: Mapping Participant B: task and gateway.	72
Figure 5.6: Mapping Participant B: end event.....	72
Figure 5.7: Example of the resulting PN for an intra-organizational BP.	72
Figure 5.8: BPMN example of the failure to satisfy Option to complete.....	73
Figure 5.9: BPMN example of the failure to satisfy Proper Completion.....	74
Figure 5.10: BPMN example of the failure to satisfy No dead activities.....	74
Figure 5.11: Unfolding a net: some steps.	76

Figure 5.12: Unfolding a net: resulting OccurrenceNet.....	76
Figure 5.13: No missing data example.....	78
Figure 5.14: Unnecessary data example.....	79
Figure 5.15: No lost updates example.....	79
Figure 5.16: Nodes of CRG.....	80
Figure 5.17: R1 eCRG.....	80
Figure 5.18: R2 eCRG.....	81
Figure 5.19: R3 eCRG.....	81
Figure 5.20: R4 eCRG.....	81
Figure 5.21: R5 eCRG.....	82
Figure 5.22: R6 eCRG.....	82
Figure 5.23: R7 eCRG.....	83
Figure 5.24: R8 eCRG.....	83
Figure 5.25: R9 eCRG.....	84
Figure 5.26: R10 eCRG.....	84
Figure 5.27: R11 eCRG.....	85
Figure 5.28: R12 eCRG.....	85
Figure 5.29: R13 eCRG: (a) Economic Office case and (b) Urban Office case	86
Figure 5.30: R14 eCRG.....	86
Figure 6.1: Verification Component Architecture	87
Figure 6.2: Sequence Diagram: Platform Verifier Scenario 1.....	89
Figure 6.3: Sequence Diagram: Platform Verifier Scenario 2.....	90
Figure 6.4: Sequence Diagram: Platform Verifier Internal Formal Scenario 1.....	91
Figure 6.5: Sequence Diagram: Platform Verifier Internal Metrics Scenario 1.....	92
Figure D.1: Elements of the control flow perspective.....	109

Figure D.2: Elements of the interaction perspective..... 109

Figure D.3: Elements of the time perspective..... 110

Figure D.4: Elements of resource and data perspective..... 110

1 Introduction

1.1. Deliverable Purpose

In the context of the Learn PAd project the models and contents quality assessment work-package intends to provide support for quality assessment. Among the other objectives, WP4 aims to analyze Business Process (BP) models based on formal verification techniques to discover possible structural problems resulting from the model design-phase.

This Deliverable presents the first results of WP4 related to the following task (from the Learn PAd DoW).

- **Task 4.1: Formal Verification of Business Processes** - Specification of BP models can be a quite complex task when many different PA offices are involved or data dependencies are considered. The result can be BP models missing to satisfy relevant structural properties (e.g. deadlock free) [92]. On the other side intra-office BP modeling requires to check that internal process specifications conforms with related roles in global specifications. This task will investigate on proper formal verification approaches to assess the satisfaction of relevant properties by defined BP models. Defined approaches will take into account specific peculiarities of the defined modeling notations, permitting to optimize well know verification strategies, as it has been proposed in [22]. The task aims at introducing proper formal verification techniques within the Learn PAd platform, in order to help BP modelers in defining global and local BP specifications.

Therefore, the main purpose of this Deliverable is to describe the quality assessment strategies, that can be implemented by different roles, which will be included within the Learn PAd platform. The implementation will be realized taking into account BP models' external qualities, such as understandability, and correctness, with reference to BP models represented using the Business Process Model and Notation (BPMN) standard¹. We also asses the application of measures (number of nodes, connector heterogeneity, etc) which provide information about the model. These measures are used to support the BP models' quality. Understandability is critical to construct new knowledge as an outcome of the PA employees learning process. It is stated in [77] that BP Models with poor results for understandability, also imply poor learnability. Than, process correctness is introduced to ensure that models are consistent, contain pathways to completion and avoid deadlocks issues. In conclusion, this Deliverable introduces the results of the design activity related to the Learn PAd platform model verification component, including both understandability guidelines and correctness analysis.

¹In the following we will use BPMN or BPMN 2.0 interchangeably to refer to version 2.0 of the notation (Release Date: January 2011); to refer to version 1.0 we write BPMN 1.0

1.2. Related Deliverables

The deliverable has been organized according to the first results of the Learn PAd project as stated by the following Deliverables.

- **D1.1. Requirements Report.** It identifies Learn PAd as a socio-technical ecosystem that is based on fundamentals of process-oriented learning and consists of a set of software components, the so-called Learn PAd platform.
- **D3.1. Domain Analysis of Business Processes in Public Administrations.** It analyzes the modeling notations and the practices in order to define a first list of concepts suitable to define the Learn PAd meta-model.
- **D3.2. Design and Initial Implementation of metamodels for Describing Business Processes in Public Administrations.** It presents the initial Implementation of a meta-model for Describing Business Processes in Public Administrations.
- **D8.1. Demonstrators BP and Knowledge models.** It presents demonstrators according to scenarios running in Public Administrations used to validate the proposed solutions.

1.3. Deliverable Structure

The deliverable is organized as follows.

- Chapter 2 introduces Learn PAd roles and quality assessment strategies for BP model.
- Chapter 3 describes some details related to the state of the art both in the area of BP model understandability and correctness.
- Chapter 4 relates to the goal of reaching a process model that can be understood by people.
- Chapter 5 discusses the need to formally check correctness for BP models.
- Chapter 6 introduces the software quality assessment mechanisms included in the Learn PAd platform.
- Chapter 7 reports some conclusions and future development.

2 Quality Assessment Strategy for Business Process Models

This chapter describes the process envisioned for the quality assessment of Business Process models – referred in the following as BP models or simply, models. Overall, the process can be partitioned into two complementary quality assessment strategies: an **automated** quality assessment strategy, and a **crowd-based** quality assessment strategy. The former, more software-intensive, employs formal model verification – as described in Chapter 5 – and automated model understandability assessment – according to the guidelines described in Chapter 4. The latter, more human-intensive, employs the feedback of the learners to improve the quality of the BP models, and, in the long term, to provide additional understandability guidelines to plug in the Learn PAd platform.

2.1. Roles and Tasks

The two envisioned strategies jointly operate to improve the quality of the models; it is therefore useful to present them as part of a single quality assessment process. Let us first consider the roles involved within the quality assessment process, and the expected tasks of each role, by looking at Fig. 2.1. Circles represent roles, while boxes represent tasks to perform. An arrow connects a role to a task, in case the role is supposed to perform such task.

Let us give a brief overview of roles and tasks. The **Modeller** first designs a model for a process, performs automated validation and then generates the Wiki pages to be used by the Learners. The **Learners** can provide feedback on the models reported in the Wiki pages. Meanwhile, the **Content Manager** monitors such feedback, and provide the Modeller with recommendations for improvement. When the Content Manager sees that some common model defects could be addressed through automated quality assessment, he contacts the **Guidelines Manager** who will take care of updating the Learn PAd platform with novel guidelines. In the following section, we give the details of each role and each task, taking Fig. 2.1 as reference.

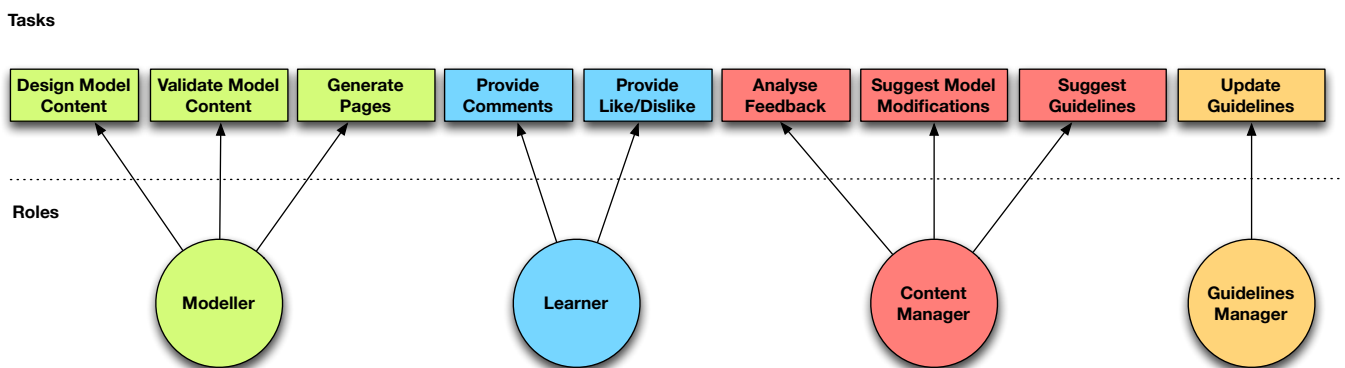


Figure 2.1: Roles and Tasks associated to the quality assessment strategy for BP Models.

The roles involved in the quality assessment process, together with their tasks, are the following.

- **Modeller.** This role is played by an expert in business process modelling. The tasks of this role with respect to model quality assessment are following reported.
 - 1) **Design Model Content:** the Modeler designs the models through the modelling platform. This task might be performed with the support of other process experts of the organization for which the model is developed. This task includes also the update of the models after validation, or after suggestions for model modifications provided by the Content Manager (described later in this section).
 - 2) **Validate Model Content:** validation of the models by means of automated quality assessment. Such validation includes both formal verification (Chapter 5) and model understandably assessment (Chapter 4). The largest part of this deliverable will focus on this specific task.
 - 3) **Generate Pages:** automated generation of Wiki pages to be loaded as learning content in the Learn PAd platform, as described in Deliverable D5.1, Sect. 5.2.

- **Learner.** This role is played by a civil servant of the organization for which the model has been developed. The tasks of this role, in the context of model quality assessment, are reported below.
 - 1) **Provide Comments:** as described in Deliverable D5.1, Learn PAd gives Learners the possibility to provide comments to improve the models (referred as annotations in Deliverable D5.1, Sect. 6). Such comments might be recommended corrections on the models, suggested according to the Learner's daily experience and practice. Moreover, such comments might be requests for clarification of the models, in case the learner does not understand the model content.
 - 2) **Provide Like/Dislike:** by means of Like/Dislike buttons, as typical of social networks, the Learner can provide an easy feedback on the quality of the models.

In the following, we will refer to Comments and Like/Dislike with the term Feedback.

- **Content Manager.** This role is played by a person who is expert in the specific process described by the BP model. A Content Manager is associated to one or more BP models of an organization. In principle, the role of Content Manager can be played by the same person who covers the role of Modeler. The Content Manager takes care of the *maintenance* of the learning content, which includes both the BP Models, and the Natural Language (NL) Content associated to the models. Since the quality of the NL Content is the topic of Deliverable 4.2, we will not stress the role of the Content Manager on the quality of the NL Content, but we will focus solely on his tasks in the quality assessment of BP models. In this sense, the tasks of this role are the following.
 - 1) **Analyze Feedback:** the Content Manager monitors the comments provided by the learners for the models he is in charge of. By reading such comments, he is able to evaluate the required improvements on the models. Moreover, by monitoring the number of Like/Dislike on a specific model, he can understand which are the models that require major improvements according to the community, and/he can prioritize updates on the models.
 - 2) **Suggest Model Modifications:** after analyzing the feedback of the Learners, the Content Manager is able to suggest modifications on the models. In this case, he will contact the Modeller, and will interact with him to update the model. Of course, if the Content Manager is played by the same person who covers the role of Modeller, then this interaction does not need to take place.
 - 3) **Suggest Guidelines:** after monitoring a set of models, the Content Manager understands that *specific* guidelines can be provided to address, at the modeling stage, some common negative feedback coming from the Learners. For example, if many Learners encounter

difficulties in understanding a specific BPMN construct, the Content Manager will recommend not to use that specific construct. The recommendations given by the Content Manager will be collected by the Guidelines Manager.

- **Guidelines Manager:** this role is covered by a person who is in charge of maintaining the Learn PAd platform. The Guidelines Manager is associated to multiple Content Managers, possibly belonging to different organizations, who will refer to him as the collector of guidelines recommendations. The main task of this role is following reported.

- 1) **Update Guidelines:** after receiving guidelines recommendations from the Content Managers, he will decide the guidelines that to plug in the Learn PAd platform for providing automated quality assessment. Given that the guidelines derive from the experience of the Learners, we expect such guidelines to be mainly related to the understandability of the models. Therefore, such guidelines will be added to the understandability guidelines described in Chapter. 4.

2.2. Quality Evaluation Process

Let us now put all the roles and the tasks together to see how the quality evaluation process operates. To this end, we will refer to Fig. 2.2. In this figure, the tasks have incoming and out-coming arrows. An arrow goes from a role to a task when such role is expected to perform such task. An arrow goes from a task to a role when the product of the task is used by the role.

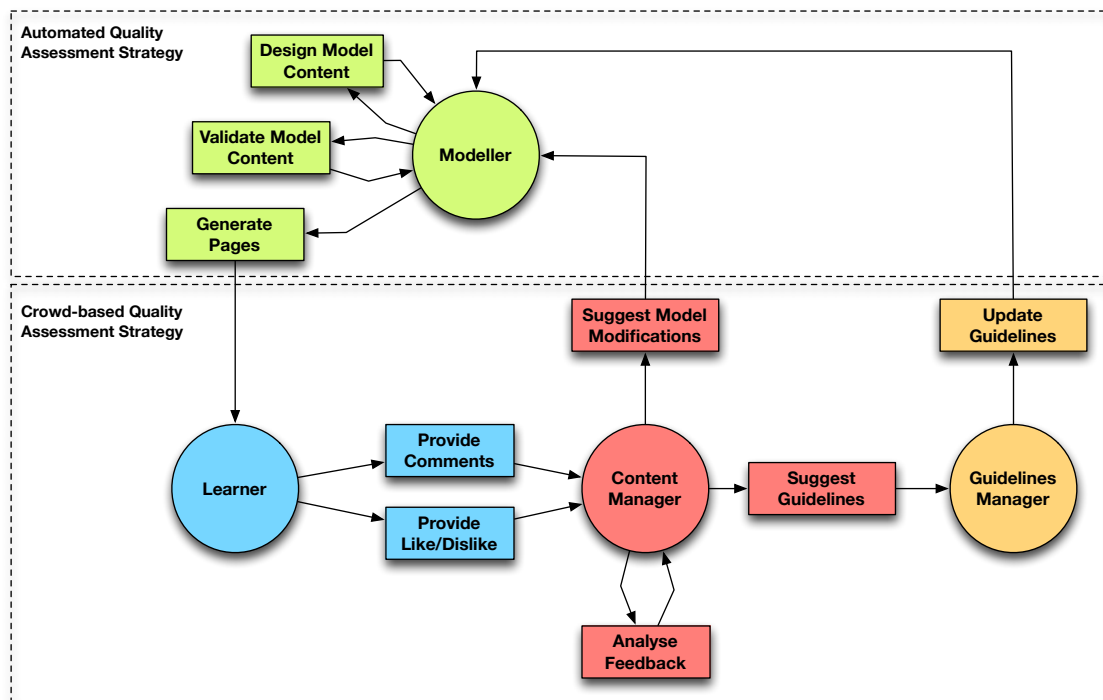


Figure 2.2: The two quality assessment strategies for BP Models depicted as components of an overall quality assessment process.

2.2.1. Automated Quality Assessment Strategy

The Automated Quality Assessment Strategy involves only one role, namely the Modeller. This role designs the model (Design Model Content) and then he performs formal verification and automated understandability evaluation, to automatically validate the produced model (Validate Model Content). In case the model does not result valid, he will update the model according to the result of the validation performed. He will iterate the validation and update the process, until the model is considered valid. When the model results valid (i.e., all the formal verification checks are passed, and all the understandability guidelines are satisfied), the Modeler will generate the Wiki pages (Generate Pages) which afterwards, will be used by the Learners. The details of the formal verification and automated understandability evaluation are reported in Sect. 5 and 4, respectively.

2.2.2. Crowd-based Quality Assessment Strategy

The Crowd-based Quality Assessment Strategy involves three roles, namely the Learner, the Content Manager, and the Guidelines Manager. The Learner provides feedback on the models, by means of comments (Provide Comments) and like/dislike (Provide Like/Dislike) buttons. The Content Manager will monitor the contributions of the Learners, and will evaluate all these feedback (Analyze Feedback) to understand and prioritize the required modifications on the model. Then, he will recommend such modifications to the Modeler (Suggest Model Modifications), who will modify the models and will repeat the Automated Quality Assessment Strategy, as described in Sec. 2.2.1. In the long term, the Content Manager will be able to identify typical weaknesses of the models for which he is in charge, according to the feedback of the users. To address these common weaknesses, he will recommend modeling guidelines to plug into the Learn PAd platform (Suggest Guidelines). The Guidelines Manager will collect guidelines recommendations from multiple Content Managers, and will define techniques to automatically assess such guidelines (Update Guidelines). These iterations will enable a refinement of the Automated Quality Assessment Strategy.

To have a practical view of how the crowd-based strategy will work, it is useful to refer to the mock-ups of the following figures. In Fig. 2.3, we see a sample Wiki page that shows a process for getting the reimbursement for expenses in a generic organization. Two types of reimbursement are foreseen, namely documented reimbursement and daily allowance reimbursement. Both types require to get a reimbursement module, which is not depicted in the model. One of the Learners provides a comment, asking about such module. The Content Manager replies to the comment, to help the Learner, and will contact the Modeler to ask him to update the model.

The screenshot shows a web browser window titled "LearnPAD" with the URL "http://LearnPAD.eu/Reimbursement". The main content is a BPMN diagram for a reimbursement process. The process starts with a start node leading to a "Get Module" activity. This leads to a decision diamond labeled "Type of reimbursement?". From this diamond, two paths emerge: one labeled "documented" leading to a "Write Expenses" activity, and another labeled "daily allowance" leading to a "Report Days" activity. Both "Write Expenses" and "Report Days" lead to a "Sum Expenses" activity. This activity leads to another decision diamond, which then leads to a "Provide Claim" activity, ending at a final node. Below the diagram are thumbs-up and thumbs-down icons with the number "34" and "10" respectively. Below the diagram is a "Summary" section with the text: "This process describes how to get reimbursement for mission expenses." Below that is a "Motivation" section: "Employees shall be reimbursed after they have performed their mission." Below that is a "Description" section: "Two types of reimbursement are foreseen by this process. The first type is documented reimbursement. The second type is reimbursement through daily allowance." Below the description are thumbs-up and thumbs-down icons with the number "0" and "30" respectively. Below the description is a comment thread with three messages: "Isn't there a module for reimbursement?", "You should download it from [http://LearnPAD/ReimbursementModule](\"http://LearnPAD/ReimbursementModule\"). The model will be updated soon.", and "Great, thanks a lot!".

Figure 2.3: Example of crowd-based quality assessment of the models. The Learner highlights a missing document from the model, and the Content Manager provides him an early feedback. Then, he will contact the Modeller to update the model.

The Modeller updates the model as in Fig. 2.4 (for simplicity, in the figure we do not report the NL Content). As we can see, the Modeler replicates the “Module” object, to ensure consistency. However, this notation decreases the clarity of the model itself, and the dislikes of the Learners increase consistently. Moreover, one of the Learners provide a comment to ask whether the “Module” object is the same or not.

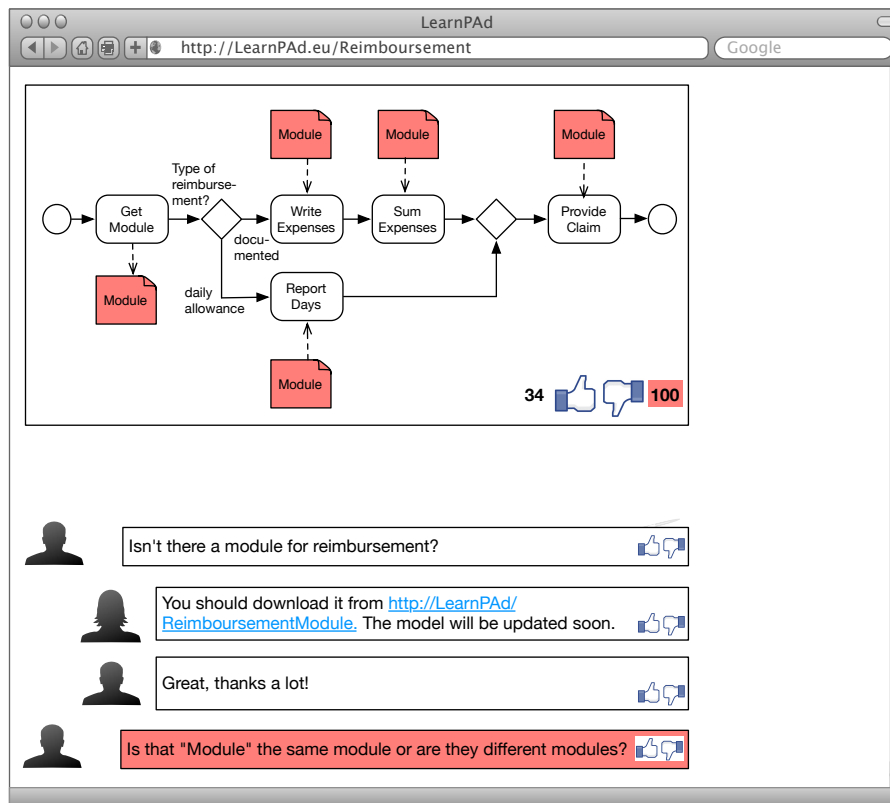


Figure 2.4: Example of crowd-based quality assessment of the models. The Modeler updates the model according to the request of the Content Manager.

The Content Manager understands that, since the “Module” object is taken as input by all the tasks, it should not be replicated. Therefore, he first contacts the Modeler to re-update the model, as shown in Fig. 2.5. Then, he contacts the Guidelines Manager to suggest the following novel guideline to be automatically verified by the platform:

Recommended Guideline: “The modeler should not replicate a document object in a model, when the document object is the only document involved in the process, and it is used by all the tasks of such process.”

The Guidelines Manager will take care of formalizing the guideline, according to the structure presented in Sec. 4. Moreover, he will take care of implementing an automated verification component for such guideline to be plugged in the Learn PAD platform.

It is worth highlighting that the crowd-based quality assessment strategy described in this section is highly human-intensive, and relies on the capabilities of the Learn PAD collaborative environment, as specified in Deliverable D5.1. In a sense, this section is a guide for Learn PAD users on *how* the crowd-based quality assessment can be put into place, once the Learn PAD platform is deployed in a specific administration.

The screenshot shows a web browser window titled "LearnPAd" with the URL "http://LearnPAd.eu/Reimbursement". The main content is a flowchart illustrating a reimbursement process:

- The process starts with a circle leading to a rounded rectangle labeled "Get Module".
- An arrow points from "Get Module" to a diamond-shaped decision node labeled "Type of reimbursement?".
- From the decision node, two paths emerge:
 - One path is labeled "documented" and leads to a rounded rectangle labeled "Write Expenses".
 - The other path is labeled "daily allowance" and leads to a rounded rectangle labeled "Report Days".
- Both "Write Expenses" and "Report Days" lead to a rounded rectangle labeled "Sum Expenses".
- An arrow points from "Sum Expenses" to another diamond-shaped decision node.
- From this second decision node, an arrow leads to a rounded rectangle labeled "Provide Claim".
- The process ends with an arrow pointing to a final circle.

Below the flowchart, there is a red box labeled "Module" with an arrow pointing to the "Get Module" step. At the bottom right of the flowchart area, there is a red box with the number "110", a thumbs-up icon, a thumbs-down icon, and the number "100".

Below the flowchart is a comment section with five entries:

- User 1: "Isn't there a module for reimbursement?" (with thumbs-up and thumbs-down icons)
- User 2: "You should download it from <http://LearnPAd/ReimbursementModule>. The model will be updated soon." (with thumbs-up and thumbs-down icons)
- User 3: "Great, thanks a lot!" (with thumbs-up and thumbs-down icons)
- User 4: "Is that 'Module' the same module or are they different modules?" (with thumbs-up and thumbs-down icons)
- User 5: "It's the same. Now the model should be fine." (with thumbs-up and thumbs-down icons)

Figure 2.5: Example of crowd-based quality assessment of the models. The Modeler re-updates the model, to increase the readability.

3 Overview and State of the Art

In this chapter we introduce a state of the art representing the starting point for the definition of quality assessment strategies for BP models. It helps to focus on the works already done in the area and make aware the reader on the different techniques and solutions already available. First of all we focus on BP model understandability and then we consider formal verification approaches.

3.1. Process Model Understandability

To make a BP model understandable, reliable, and reusable it is important to ensure its quality. Several approaches that work in this direction are described in the literature. We have classified them based on their main research topic: (1) approaches focused on improving BP design through the suggestion of modeling guidelines (2) approaches which identify process model metrics to evaluate model qualities (3) approaches which establish thresholds for the identified metrics.

3.1.1. Business Process Modeling Guidelines

Here we report some of the approaches that are intended to provide advice and guidelines to improve the BP model qualities.

- Becker et. al. [6] propose a set of guidelines to improve six characteristics of a process model such as correctness, clarity, relevance, comparability, economic efficiency, and systematic design. The provided guidelines aim at improving the quality of the model creation process as well as that of the conceptual model itself. The principle of correctness thereby proposes that the real world excerpt has to be depicted correctly with respect to its content. The principle of relevance prescribes that only elements must be depicted which are relevant for the modeling purpose. The principle of economic efficiency demands that the costs for creating models must not exceed the expected utility. The principle of clarity proposes that a model has to be understandable and readable for the respective users. The principle of comparability requires that models have to be created in such a way that their content can be compared with each other. The principle of systematic design finally proposes that multiple views have to be used for the modeling of different aspects which should be adjusted to each other. Since they were first introduced, the GoM have repeatedly been refined and adjusted according to specific modeling purposes, among others for the modeling of BPs. However, they do not contain concrete measures/guideline to achieve the mentioned goals, which makes their practical application during the modeling process difficult.
- Mendling et. al. [53] study, through interviews, the understandability of models. They concluded that in addition to the factor of the basic individual knowledge, the size of the model is the dominant aspect of understandability.
- Mendling et. al. [54] define a set of seven process modeling guidelines (7PMG) (see Table 3.1) that are supposed to guide the modeler in designing understandable models that are less prone to errors. Therefore, G1 recommends to use as few elements as possible. G2 suggests to minimize the routing paths per element. The higher the degree of elements in the process model the harder

it becomes to understand the model. G3 demands to use one start and one end event, since the number of start and end events is positively connected with an increase in error probability. Following G4, the models should be structured as much as possible. Unstructured models tend to have more errors and are understood less well. G5 suggests to avoid OR routing elements, since models that have only AND and XOR connectors are less error-prone. G6 recommends using the verb-object labeling style because it is less ambiguous compared to other styles. Finally, according to G7, models should be decomposed if they have more than 50 elements.

G1	Use as few elements in the model as possible
G2	Minimize the routing paths per element
G3	Use one start and one end event
G4	Model as structured as possible
G5	Avoid OR routing elements
G6	Use verb-object activity labels
G7	Decompose a model with more than 50 elements

Table 3.1: Seven process modeling guidelines (Mendling et. al. [54])

- Reijers et. al. [68] present the SIQ framework for the quality of BP models. Its core consists of the three dimensions of syntactic, semantic, and pragmatic quality. This framework can be applied to different types of modeling artifacts (both graphic and text-oriented) such as entity relationship diagrams, data flow diagrams, object models, use case descriptions, etc. The pragmatic quality refers to process models that can be understood by the stakeholders. To ensuring pragmatic quality the authors use the seven process modeling guidelines 7PMNG proposed by Mendling et. al. [54].
- Silingas et. al. [81] analyze six practical BPMN refactoring cases that are common in practice. They present original BP samples, specifying the bad smells contained, suggesting best practices/patterns to comply with, and refactoring the original versions into better quality BP models.
- Bruce Silver wrote a book [83], which highlights the use of a disciplined approach called “method and style” to help the modeler creating BPMN models that are correct, complete, and clear.
- Mendling et. al. [55] refine existing 7PMG; see Table 3.2. In particular G4 and G5 have been extended.

G1	Do not use more than 31
G2	No more than 3 inputs or outputs per connector
G3	Use no more than 2 start and end events
G4.a	Model as structured as possible
G4.b	Use design patterns to avoid mismatch
G5.a	Avoid OR-joins and OR-splits
G5.b	Minimize the heterogeneity of connector types
G5.c	Minimize the level of concurrency
G6	Use verb-object activity labels
G7	Decompose a model with more than 31 elements

Table 3.2: Ten process modeling rules (Mendling et. al. [55]).

- Bosshart et. al. [8] propose best practices for the Swiss government to standardize the graphical representation of administrative process in BPMN.

Other sources for BP modeling guidelines can be found online. In particular we consider valuable the contribution by Bruce Silver [82], the one by John Doe [19], and the web pages entitled: Modeling Best

Practices¹, BPMN Modeling Guidelines², BPMN 2.0 Best Practices³, and Best Practices in modeling⁴.

3.1.2. Business Process Model Metrics

Here we report some of the approaches based on the individuation of BP model metrics.

- Cardoso [11] proposes a Control Flow Complexity (CFC) metric and Rolón et. al. [71] present the use and validation of the CFC metric to evaluate the complexity of BP models developed with BPMN 1.0. The complexity is evaluated from a control-flow perspective. The authors conclude that CFC metric is highly correlated with the control-flow complexity of a BP and therefore with its understandability and modifiability.
- Rolón [72] defines the measures that can be applied to BPMN 1.0 models in order to quantify the understandability and modifiability of conceptual models. These measures have been validated through a correlation and regression analysis [73].
- Reynoso et. al. [70] measure the understandability of BPMN 1.0 models, but understandability is an external quality attribute that can only be measured when the models are completed. For this reason, indirect measures for understandability are needed, focusing on the structural properties of BPMN models, such as their structural complexity.
- Overhage et. al. [60] present the 3QM-Framework, an analytical approach to systematically determine the quality of BP models. The 3QM-Framework makes three contributions: it provides quality marks, metrics, and measurement procedures to quantify the quality level as elements of a theoretically justified quality model.

A list of metrics, that have been used to monitor BP Model complexity, can be found in Table A.1 in Appendix A. The firsts metrics (until the row with "...") count the number of BPMN elements and (for a matter of space) we choose to not report all of them. The following metrics are instead based on combination of BPMN elements or they have been adapted from other researching areas e.g. software complexity metrics. Similar works that report a collection of metrics can be found in literature such as [58], [37], and [38].

3.1.3. Threshold for Business Process Model Metrics

According with the defined metrics for Business Process modeling, some authors tried to identify thresholds which may indicate the level of model qualities e.g. high level of understandability (if some metrics values do not exceed the thresholds) or low level of understandability (if the metrics values exceed the thresholds). Following we report the main sources we considered from the literature. Some of them have already been cited in the previous section since the authors are also the ones that provided the metrics.

- Sanchez et. al. [75] [78] investigate structural metrics and their connection with the quality of process models, namely understandability and modifiability. They consider metrics, like the ones reported in Table 3.3. They analyzed performance measures including time, correct answers and efficiency from a family of experiments for correlations with an extensive set of structural process model metrics. Their findings demonstrate the potential of these metrics to serve as validated predictors of process model quality. Based on the results of paper they determine threshold values to distinguish different levels of process model quality; Table 3.4 reports the identified threshold values for understandability.

¹Located on the Business Process Incubator provided by the Trisotech company: <http://www.bpmnquickguide.com>

²Hosted by Signavio GmbH and located at: <http://www.modeling-guidelines.org/conventions/signavio-best-practice/>

³Provided by the Camunda company and located at: <http://camunda.org/bpmn/examples/>

⁴Provided by the Bizagi company and located at: <http://help.bizagi.com/processmodeler/en>

Measure	Description	U*	M*
Measures of Rolón [72]			
TNSF	Total Number of sequence flows	X	
TNE	Total Number of events	X	
TNG	Total Number of gateways	X	
NSFE	Number of sequence flows from events	X	
NMF	Number of message flows	X	
NSFG	Number of sequence flows from gateways	X	X
CLP	Connectivity level between participants	X	
NDOOut	Number of data objects which are outputs of activities	X	
NDOIn	Number of data objects which are inputs of activities	X	
CLA	Connectivity level between activities		X
Measures of Cardoso [11]			
CFC	Control flow complexity. Sum over all gateways weighted by their potential combinations of states after the split	X	X
Measures of Mendling [52]			
Number of nodes	Number of activities and routing elements in a process model	X	
Gateway mismatch	Sum of gateway pairs that do not match each other, e.g. when an AND-split is followed by an OR-join	X	X
Depth	Maximum nesting of structured blocks in a process model	X	
Connectivity coefficient	Ratio of total number of arcs in a process model to its total number of nodes	X	
Density	Ratio of total number of arcs in a process model to the theoretically maximum number of arcs		X
Sequentiality	Degree to which the model is constructed from pure sequences of tasks	X	X

U*: Understandability, M*: Modifiability

Table 3.3: Understandability and Modifiability Measures.

- Mendling et. al. [55] derive thresholds for a set of structural measures for predicting errors in conceptual process models. This is helpful for understanding, for example, that size and complexity are general driving forces of error probability. Significant thresholds were identified, based on ROC curves and the Area Under the Curve [32], and adapted to refine existing modeling guidelines (7PMG) in a quantitative way. The resulting threshold are reported in Table 3.5.

⁵In the original paper [75] this value is reported as 44, we believe it to be a typing error, that is why we suggest the value 14 instead.

Model Metric	Very Inefficient	Rather Inefficient	Rather Efficient	Very Efficient
N°nodes	65	50	37	31
GatewayMismatch	29	16	6	1
Depth	4	2	1	1
Coefficient of connectivity	1,7	1,1	0,6	0,4
Sequentiality	0,1	0,35	0,6	0,7
TNSF	72	49	34	20
TNE	20	12	7	2
TNG	17	10	5	0
NSFE	28	13	4	0
NMF	27	15	7	1
NSFG	40	22	11	0
CLP	7,5	4,23	2,2	0,2
NDOIN	31	14 ⁵	4	0
NDOOUT	23	11	3	0
CFCxor	30	17	8	1
CFCor	9	4	1	0
CFCand	4	2	0	0

Table 3.4: Threshold values for conceptual model metrics (Snchez et. al. [75])

Metric	Threshold
Conn. Heterogeneity	0.4
Conn. Mismatch	4.5
Token Splits	7.5
CFC	4.5
Nodes	31.5
Density	0.033
End-events	2.5
Sequentiality	0.21
Depth	0.5
Max. Conn. Degree	3.5
Coeff. Connectivity	1.021
Structuredness	0.79
Separability	0.49
Or-Sprilts	0.5
Start-Events	2.5
Av.Conn. Degree	3.09
Cyclicity	0.005
Or-Joins	0.5

Table 3.5: Thresholds identified based on ROC Curves (Mendling et. al. [55])

- Sanchez et. al. in [76] focus on identifying thresholds for gateway complexity measures such as: Control-Flow Complexity (CFC), Gateway Mismatch (GM), Gateway Heterogeneity (GH), Average Gateway Degree (AGD), Maximum Gateway Degree (MGD) and Total Number of Gateways, (TNG). The authors specially focus on the relation between those complexity measures and the understandability external quality of a model. The thresholds resulting from their experiments are presented in Table 3.6.

Threshold	Linguistic Label
<i>Control-Flow Complexity</i>	
$CFC \leq 13$	Fairly low measure value or fairly easy to understand/modify.
$13 < CFC \leq 22$	Low measure value or easy to understand/modify.
$22 < CFC \leq 37$	Medium measure value or moderately difficult to understand/modify.
$37 < CFC \leq 51$	High measure value or difficult to understand/modify.
$CFC > 51$	Fairly high measure value or fairly difficult to understand/modify.
<i>Gateway Mismatch (GM)</i>	
-	Fairly low measure value or fairly easy to understand/modify.
$GM \leq 6$	Low measure value or easy to understand/modify.
$6 < GM \leq 15$	Medium measure value or moderately difficult to understand/modify.
$15 < GM \leq 20$	High measure value or difficult to understand/modify.
$GM > 20$	Fairly high measure value or fairly difficult to understand/modify.
<i>Gateway Heterogeneity (GH)</i>	
$GH \leq 0.62$	Fairly low measure value or fairly easy to understand/modify.
$0.62 < GH \leq 0.79$	Low measure value or easy to understand/modify.
$0,79 < GH \leq 0.92$	Medium measure value or moderately difficult to understand/modify.
$0.92 < GH \leq 0.94$	High measure value or difficult to understand/modify.
$0.94 < GH$	Fairly high measure value or fairly difficult to understand/modify.
<i>Average Gateway Degree (AGD)</i>	
$AGD \leq 3.67$	Fairly low measure value or fairly easy to understand/modify.
$3.67 < AGD \leq 3.83$	Low measure value or easy to understand/modify.
$3.83 < AGD \leq 4.06$	Medium measure value or moderately difficult to understand/modify.
$4.06 < AGD \leq 4.18$	High measure value or difficult to understand/modify.
$4.18 < AGD$	Fairly high measure value or fairly difficult to understand/modify.
<i>Max. Gateway Degree (MGD)</i>	
$MGD \leq 4$	Fairly low measure value or fairly easy to understand/modify.
$4 < MGD \leq 5$	Low measure value or easy to understand/modify.
$5 < MGD \leq 7$	Medium measure value or moderately difficult to understand/modify.
$7 < MGD \leq 9$	High measure value or difficult to understand/modify.
$9 < MGD$	Fairly high measure value or fairly difficult to understand/modify.
<i>Total Number of Gateways (TNG)</i>	
$TNG \leq 9$	Fairly low measure value or fairly easy to understand/modify.
$9 < TNG \leq 12$	Low measure value or easy to understand/modify.
$12 < TNG \leq 18$	Medium measure value or moderately difficult to understand/modify.
$18 < TNG \leq 22$	High measure value or difficult to understand/modify.
$22 < TNG$	Fairly high measure value or fairly difficult to understand/modify.

Table 3.6: Threshold values and linguistic labels for gateway complexity measures (Sanchez et. al. [76]).

Some of the presented metrics and threshold will be associated to the guidelines we present in section 4. In fact, those metrics and threshold may be used to evaluate if a model is following a particular guideline or if it needs to be modified.

3.2. Business Process Formal Verification

Several application areas use Formal verification of BP model in order to increase the correctness of BP models. Different surveys have been already published in this area. [30] provides a comprehensive overview with reference to process correctness, business compliance and process variability verification while [57] provides an overview of Business Process checking. A summary of the analyzed sources can be found in Appendix B.

3.2.1. Petri Net Based Approaches

We provide an introduction to Petri Net (PN) in Appendix C while, following we report a list of the main references about PN.

- Van der Aalst [84] introduces soundness to the field of BPM by translating workflows into PNs, and Wynn et. al. [96] perfected the application by allowing Or-joins and cancellation regions. As a consequence, PNs are commonly used as intermediate formalism by soundness verification frameworks, including in [88] where van Dongen et. al. uses them to verify EPC.
- Dehnert, Juliane, and Peter Rittgen [15] present a pragmatic approach to correctness which only requires that the PN represents some valid behavior with reference to relaxed soundness property. Few years later, Dehnert and van der Aalst [16] present a methodology to bridge the gap between BP modeling and workflow specification. The methodology consists of five steps and is illustrated using event-driven process chains as a BP modeling language and PNs as the workflow specification language.
- Van der Aalst [85] uses PNs to specify the partial ordering of tasks. Founded on a Petri-Net-based representation of the workflow, the paper tackles the problem of verification and it provides techniques to verify the so-called soundness property. The discussion has been also extended to Task Structures [86]. Verbeek [91], introduced details relate to the functionality of Woflan analysis tool with reference to a travel agency example.
- Finally, the notion of soundness has been extended toward separability with reference to the Workflow Net BP modeling approach [89].

3.2.2. Model Checker Input Language

Another popular method to verify BP properties is translating processes into a model checker input language.

- Masalagiu et. al. [49] verify BPMN 1.0 by translating it (via a Petri Net intermediate model) into the model checker input language TLA+. Karamanolis et. al. [35] translate processes to the process algebra FSP and check the result with the Labeled Transition System Analyzer. The model-checking techniques associated with TRACTA can be used to check a workflow system exhaustively, against both generic (deadlock, safety and liveness) and domain-specific properties. Koehler et. al. [43] translate a BP model in a corresponding IT model based on non-deterministic automata with state variables; it is the nuSMV input language.
- Nakajima [59] proposes a method to extract the behavioral specification from a BPEL application program and to analyze it by using the SPIN model checker. Wong and Gibbons [93] demonstrate

how process algebra, such as CSP, can be applied to model complex workflow systems. The same authors provided a formal semantics for BPMN 1.0 thanks to the expressibility of CSP and Z notation [94], and use it to formally check compatibility of BPMN 1.0 processes [95]. Nevertheless, the approach does not introduce a user-friendly and integrated environment for verification purposes. Puhlmann presents the foundations of a tool for static analysis of BP models based on a mapping from a subset of BPMN to π -Calculus [63] [62]. Borger and Thalheim [7] define an extensible semantical framework for Business Process modeling notations using the Abstract State Machine method.

- Janssen et. al. [34] propose a manager oriented solution for Business Process Verification. Spin is used as model checker underlying a graphical modeling language, and requirements are specified using business requirements patterns (sequences, consequences, combined occurrence or exclusion, required precedence of activities), translated to LTL. Finally, Mallek et. al. [48] introduce a verification of interoperability requirements. This paper focuses on application of the model checker UPPAAL, and entails translating the collaborative process model from BPMN 1.0 to Network of Timed Automata; interoperability requirements are also formalized in the temporal logic TCTL.

3.2.3. Business Process Compliance

A dominant number of compliance frameworks focuses on verifying imperative specifications such as BPMN, BPEL, EPC, and UML sequence diagrams.

- Anderson et. al. [1] explicate how model-checking technology can aid in the design and assurance of e-business processes in complex digital environments. Mauw et. al. [50] present instead an application of the SPIN model-checker in Testbed, a framework for BP reengineering. Business processes are described using the AMBER language and then translated into a state machine description in PREMOLA, which is the input language of the SPIN model-checker. The correctness properties concerning the behavioral aspects and the data entities used in the specification are checked on the resulting PROMELA program using SPIN.
- Arbab et. al. [2] describe how BPMN 1.0 diagrams can be represented by means of a semantically precise channel-based coordination language called Reo which admits formal analysis using model checking and bisimulation techniques.
- Awad et. al. [4] introduce an approach for automated compliance checking. Compliance rules are specified in BPMN-Q (a query language that they developed) and translated into temporal logic formulae that serve as input to model checkers which in turn verify whether a process model satisfies the requested compliance rule. They also apply a set of reduction rules to address the problem of state-space explosion. Awad et. al. [5] improve the previous work, not focusing only on the verification of the control flow but, incorporating the data perspective into the specification of compliance rules. They do so extending BPMN-Q to express those rules and mapping them into PLTL. In addition they provide a visualization of the paths that violates compliance rules.
- Goedertier and Vanthienen [26] investigate the use of temporal deontic assignments on activities as a means to declaratively capture the control-flow semantics that reside in business regulations and business policies. In particular, they introduce PENELOPE, a language to express temporal rules about the obligations and permissions in a business interaction, and an algorithm to generate compliant sequence-flow-based process models that can be used in BP design.
- Ghose and Koliadis in [25] deal with BPMN 1.0 processes that are further refined and represented in a form of semantically-annotated digraphs called Semantic Process Networks (SPNets). Compliance rules in this work are modeled using Computation Tree Logic (CTL).

Some works especially focus on artifact-centric process models. Gerede and Su [24], propose a specification language ABSL based on computation tree logic to specify artifact behaviors in artifact-centric process models while Deutsch et. al. [17] propose verifying the compliance of artifact-centric processes against properties expressed in an extension of LTL.

- Several other approaches consider specific categories of compliance rules. For example, Governatori et. al. [28] developed a Formal Contract Language (FCL) for representing compliance requirements extracted from service contracts. Sadiq et. al. [74] and Governatori et. al. [27] use FCL, respectively, to encode compliance rules regarding a purchase-to-pay scenario and an account opening process in private banking. Then, the authors continue by extending this framework, in [29], with goals to provide a fully declarative description. Ly et. al. [47] present an approach that enables the instantiation and verification of process-independent compliance rules over process models using domain models.
- Montali et. al. [56] discuss the static verification of declarative Business Processes. They propose a framework based on the ConDec graphical notation for modeling Business Processes, and on Abductive Logic Programming technology for verification of properties. Pulvermuller et. al. [64] aim at verifying the compliance of design-time EPC using an extension of CTL that differentiates between events and functions.
- Knuplesch et. al [39] deal with issues related to BP compliance and encode them using LTL; Knuplesch and Reichert [40] instead, presented two ways for modeling compliance rules: LTL and CRGs.
- Knuplesch et. al. [42] and Semmelrodt et. al. [80], provide an approach that extends visual compliance rule languages with the ability to consider data, time, resources, and partner interactions when modeling BP compliance rules. Those extensions are introduced as part of extended compliance rule graphs (eCRG), which are based on the compliance rule graph (CRG) language.
- Knuplesch and Reichert [41] introduce an operational semantics for the extended compliance rule graph (eCRG) language which allow to detect compliance violations at run-time and visually highlights their causes. Moreover, it allows providing recommendations to users in order to proactively ensure for a compliant continuation of a running BP.
- Fellmann M. et. al [23] proposes a survey on BP compliance which comprehends also other relevant works.

4 Models Understandability

The Learn PAd project considers BP models, described using BPMN, as valuable resources for the representation of Public Administrations services. In particular, within Learn PAd project, BP models are considered fundamental in the process of learning about Public Administration activities in a user-centric perspective. BP models contributes to facilitate the process of learning of a Public Administration employee, hence we agree that BP model understandability plays a main role in the matter. This section relates to the goal of reaching, through the use of guidelines, a BP model that can be understood.

4.1. Understandability Guidelines

It is to guarantee BP Model understandability that, we collected, refined, and elaborated guidelines that a modeler should follow for modeling BPs. For the modeling guidelines we referred to multiple sources, that we already mentioned in 3, which include: Scientific papers, books, online articles and webpages. In this section we propose the list of guidelines divided in different categories.

4.1.1. Categories

We divided the guidelines in categories which name reflects the main characteristic of the guideline. However, the categories are not really strict; it may happen that a guideline has characteristics which belong to one or more categories at the same time. In our case we decided to group such guidelines based on their main scope. The categories are reported in the following.

- **General:** it refers to general rules that impact on different aspects of the process model.
- **Notation Usage:** it refers to best practices in the usage of the BPMN Syntax.
- **Labeling:** it refers to the correct use of names/labels, assigned to BPMN elements.
- **Patterns:** it refers to patterns that may be applied during the modeling.

In Table 4.1 we present a template of the tables used to describe the guidelines. In particular

- *Guideline name:* it represents the name of the guideline.
- *Guideline id:* it is a number that represents the id of the guideline.
- *Description:* description of the guideline.
- *Convention concerning the name:* if present, it concerns guideline for labelling elements.
- *Symbol:* if the guideline concerns a BPMN element, here the symbol for the BPMN element is reported.
- *Source:* if present, it indicates the origin of the guidelines otherwise it has been added to this context.

- *Associated Metrics and Thresholds*: if present, it indicates the metrics and thresholds associated to the guideline, if the result of metrics is 0 the model is compliant to the guidelines.
- *Bad/Good modeling*: graphical representation of bad and good practice.

Based on the guidelines category, the template may not contain all the fields describes above. In the category *General*, fields like: symbol, bad/good modeling, convention concerning the name, are not reported since considered unnecessary.

Guideline Name		Guideline ID
Name		ID
Description		Symbol of the element
The diagram describes the entire process		
Convention concerning the name		
BPMN diagrams are always marked with a noun + a verb endlessly.		
Source		
Origin of the guidelines.		
Associated Metrics		
Metrics of guideline.		
Convention on the modeling		
Bad Modeling	Good Modeling	

Table 4.1: Template of description guidelines of Business Process Model Notation.

4.1.2. General Guidelines

In this section we present general guidelines, that do not refer to specific BPMN elements.

Follow Guidelines

Guideline Name	Guideline ID
Follow Guidelines	1
Description	
The modeler should follow guidelines which aim to guarantee the modeling of business process models which are as understandable as possible.	

Consistent use of guidelines

Guideline Name	Guideline ID
Consistent use of guidelines	2
Description	
Consistency is essential: when the modeler adheres to a set of guidelines, he/she should apply them toward the entire modeling phase.	

Use a proper software tool

Guideline Name	Guideline ID
Use a proper software tool	3
Description	
For a good process representation choose a proper BPMN tool. Several tools can be chosen for Business Process Modeling, and the meaning of the diagram does not change from one tool to the another. However, even though BPMN is a standard, the tools are not all equally good. Some of them can produce diagrams containing the standard shapes and connectors, but they do not “understand” their meaning. They cannot, for example, validate the model, or save it in XML interchangeable with another BPMN tool.	

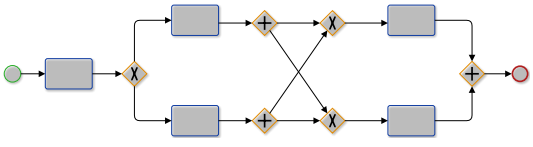
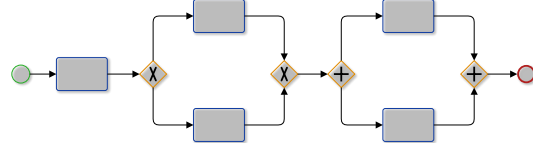
Validate the model

Guideline Name	Guideline ID
Validate the model	4
Description	
The models should comply with the BPMN standard. Once the process logic has been defined, validate the model ensuring the correct use of BPMN elements. Several BP modeling tools allow to automatically validate the model.	
Source	
[83]	

Apply hierarchical structure with SubProcesses

Guideline Name	Guideline ID
Apply hierarchical structure with SubProcesses	5
Description	
The modeler should create a hierarchical Business Process Model with multi layers of details for the Process. BPMN sub-processes are used to split the Process into “phases” or “layers”. The modeler can expand the sub-processes later to expose details of lower levels of hierarchy. A process will contain multiple pages, but internally the integrity of a single model is maintained. Moreover the modeler should pay attention to avoid cycles that may be developed through false/double linkage.	
Source	
[83]	

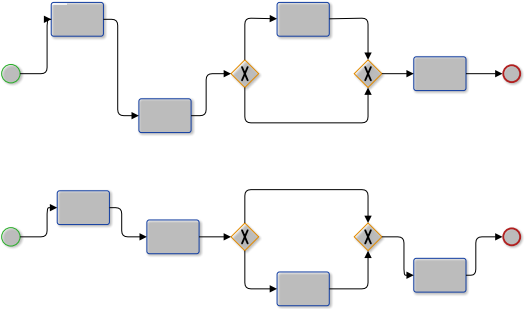
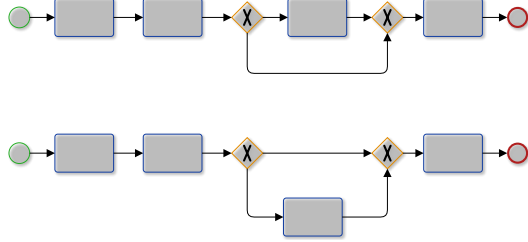
Apply symmetric modeling

Guideline Name	Guideline ID
Apply symmetric modeling	6
Description	
<p>The modeler should model as structured as possible. Symmetric structures increase understandability of BPMN process models - for both experienced and inexperienced BPMN users. Well-structuredness, means that for every node with multiple outgoing arcs (a split) there is a corresponding node with multiple incoming arcs (a join), such that the set of nodes between the split and the join form a single-entry-single-exit (SESE) region.</p>	
Source	
[54]	
Convention on the modeling	
<p>Bad Modeling</p> 	<p>Good Modeling</p> 

Minimize model size

Guideline Name	Guideline ID
Minimize model size	7
Description	
<p>The modeler should try to keep models as small as possible for example by using Subprocesses, to split the process in phases, or using Call Activities to re-use other processes. Large process models are difficult to read and comprehend. Additionally, they tend to contain more errors. Defining the correct scope of tasks and level of detail of processes is key to reduce the overage of information.</p>	
Source	
[54, 55]	
Associated Metrics and Thresholds	
$MinimizeModelSize(x) = \begin{cases} 0 & \text{if } SN \leq 31 \\ 1 & \text{otherwise} \end{cases}$ <p>where: $x \in \text{Nodes of BPMN Model} \wedge$ SN is the number of nodes: number of activities and routing elements in a process model.</p>	

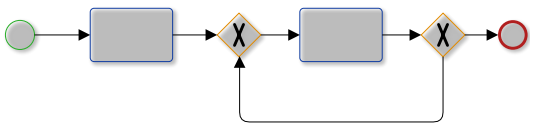
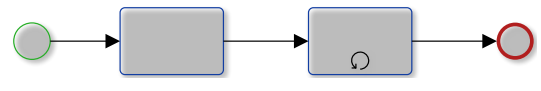
Highlight the “happy path”

Guideline Name	Guideline ID
Highlight the “happy path”	8
Description	
The modeler should make the process logic visible in the model. The “happy path” - a sequence of activities that will be executed if everything goes as expected without exceptions - should be easily identified when reading a model. Model the happy path first and then the alternative flows.	
Source	
http://www.bpmnquickguide.com/viewit.html http://help.bizagi.com/processmodeler/en/index.html?best_practices_in_modeling.htm	
Convention on the modeling	
Bad Modeling 	Good Modeling 

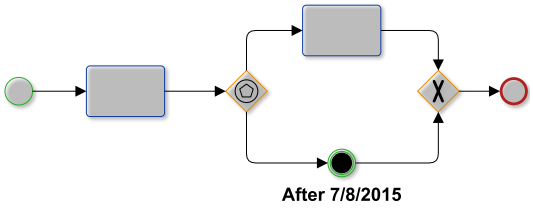
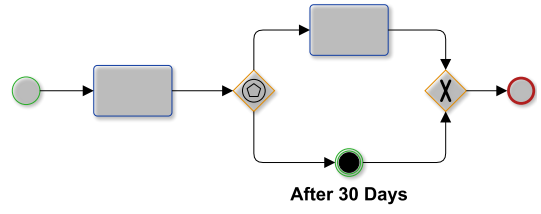
Minimize concurrency

Guideline Name	Guideline ID
Minimize concurrency	9
Description	
The modeler should minimize the level of concurrency which means to reduce the use of parallel gateways and ad-hoc subprocesses. Concurrency, which is represented by parallel gateways, may generate ambiguity, especially if the activities in parallel are “manual tasks” and only one person is responsible for those. In this case there will be no parallelization but it is up to the person to decide the tasks execution order.	
Source	
[54, 55]	

Model loops explicitly

Guideline Name	Guideline ID
Model loops explicitly	10
Description	
The modeler should explicitly model a loop, via tasks or via subprocesses marking them with the loop marker.	
Convention on the modeling	
<p>Bad Modeling</p> 	<p>Good Modeling</p> 

Set generic timers

Guideline Name	Guideline ID
Set generic timers	11
Description	
The modeler should avoid specific date and time conditions as they inhibit the re-usability of the process.	
Convention on the modeling	
<p>Bad Modeling</p> 	<p>Good Modeling</p> 

Activity Description

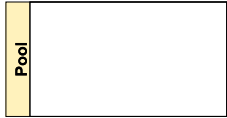
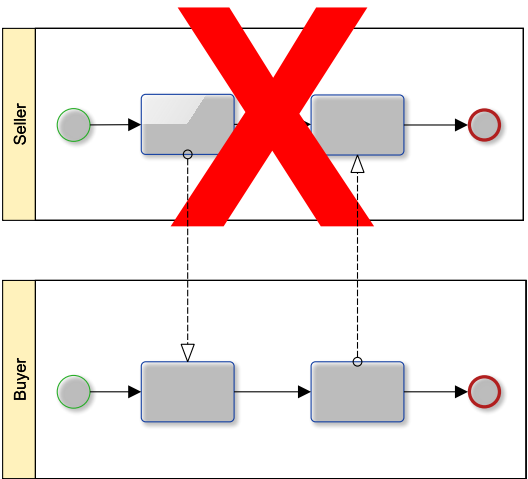
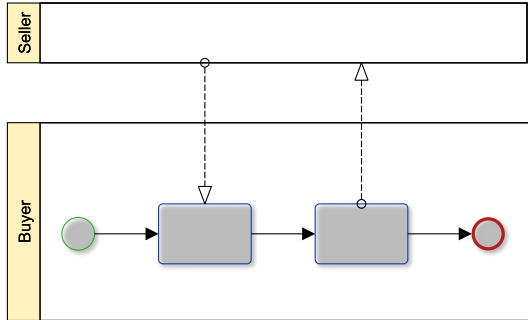
Guideline Name	Guideline ID
Activity Description	12
Description	
The modeler should provide a brief description for each activity in the model.	

Minimize gateway heterogeneity

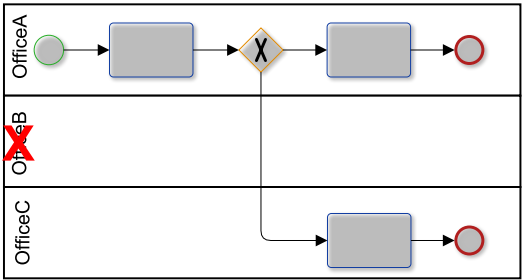
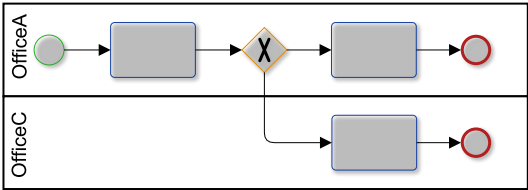
Guideline Name	Guideline ID
Minimize gateway heterogeneity	13
Description	
The modeler should minimize the heterogeneity of gateway types. The use of several type of gateway may cause confusion against the simplicity of using few main type of gateways.	
Source	
[54, 55]	
Associated Metrics and Thresholds	
$\text{MinimizeGatewayHeterogeneity}(x) = \begin{cases} 0 & \text{if } GH \leq 0.92 \\ 1 & \text{otherwise} \end{cases}$ <p>where: $x \in \text{Gateways} \wedge GH$ is the Gateway Heterogeneity.</p>	

4.1.3. Notation Usage Guidelines



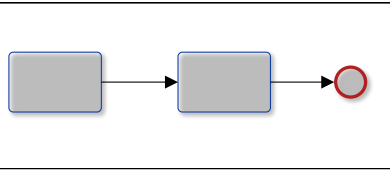
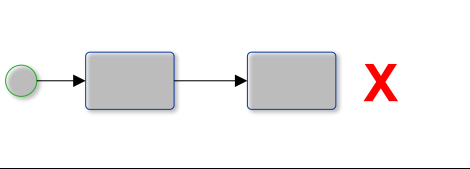
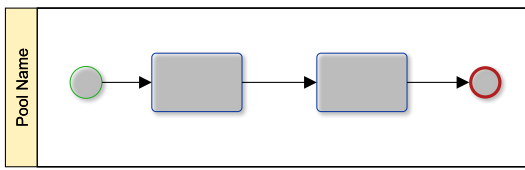
Consistent usage of pools

Guideline Name	Guideline ID
Consistent usage of pools	14
	
Description	
<p>The modeler should define as many pools as processes and/or participants. Use a black-box pool to represent external participant/processes. The usage of pools is necessary; it is important to design process models consistently and to define process owner exactly. Moreover the modeled pools need to be in correlation with each other and have to be linked to the main process through message exchange.</p>	
Source	
[8, 19]	
Convention on the modeling	
<p>Bad Modeling</p> 	<p>Good Modeling</p> 


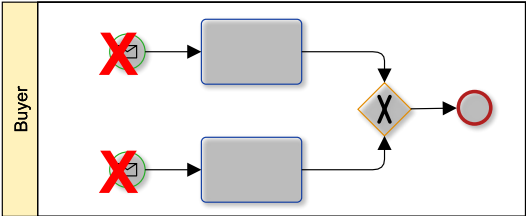
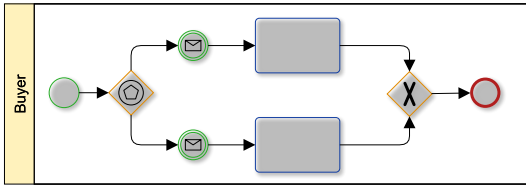
Consistent usage of lanes

Guideline Name	Guideline ID
Consistent usage of lanes	15
	<div style="border: 1px solid black; padding: 5px; width: fit-content;">Lane</div>
Description	
<p>The modeler should model internal organizational units as lanes within a single process pool, not as separate pools; separate pools imply independent processes. Create a lane, in a pool, only if at least one activity or intermediate event is performed in it.</p>	
Source	
[8, 19]	
Convention on the modeling	
<p>Bad Modeling</p> 	<p>Good Modeling</p> 


Explicit usage of start and end events

Guideline Name	Guideline ID
Explicit usage of start and end events	16
	  <small>Start Event End Event</small>
Description	
<p>The modeler should explicitly make use of start and end events. The use of start and end events is necessary to represent the different states that begin and complete the modeled process. Processes with implicit start and end events are undesirable and could lead to misinterpretations.</p>	
Source	
[8, 19, 54, 55]	
Associated Metrics and	
$ExplicitStartEndEvents(x) = \begin{cases} 0 & \text{if } TNSE > 0 \wedge TNEE > 0 \\ 1 & \text{otherwise} \end{cases}$ <p>where: $x \in \text{Events} \wedge$ $TNSE$ is the total number of start events \wedge $TNEE$ is the total number of end events</p>	
Convention on the modeling	
<p>Bad Modeling</p> <div style="border: 1px solid black; padding: 5px; margin-bottom: 5px;"> X  </div> <div style="border: 1px solid black; padding: 5px;">  </div>	<p>Good Modeling</p> <div style="border: 1px solid black; padding: 5px; text-align: center;">  </div>

Consistent usage of start events

Guideline Name	Guideline ID
Consistent usage of start events	17
	
Description	
<p>The modeler should include, in the model, only one start event. Where necessary, alternative instantiations of the process should be depicted with separate start events and using a event-based start gateway.</p>	
Source	
<p>[8, 19, 54, 55]</p>	
Associated Metrics and Thresholds	
$ConsStartEvents(x) = \begin{cases} 0 & \text{if } TNSE = 1 \\ 1 & \text{otherwise} \end{cases}$ <p>where: $x \in Events \wedge$ $TNSE$ is the total number of start events</p>	
Convention on the modeling	
<p>Bad Modeling</p> 	<p>Good Modeling</p> 

Consistent usage of end events

Guideline Name	Guideline ID
Consistent usage of end events	18
	 End Event

Description

The modeler should distinguish success and failure end states in a process or a sub-process with separate end events. Flows that end in the same end state should be merged to the same end event. Therefore, separate end events that do not represent distinct end states must be merged in a single end event.

Source

[8, 19, 54, 55]

Associated Metrics and Thresholds

$$ConsStartEvents(x) = \begin{cases} 0 & \text{if } NENE \leq NEMsE \leq NEEE \leq \\ & NECaE \leq NECoE \leq NELE \leq \\ & NEMuE \leq NETE \leq 1 \\ 1 & \text{otherwise} \end{cases}$$

where:

$x \in \text{Events}$

$NENE$ is the number of End None Events

$NEMsE$ is the number of End Message Events

$NEEE$ is the number of End Error Events

$NECaE$ is the number of End Cancel Events

$NECoE$ is the number of End Compensation Events

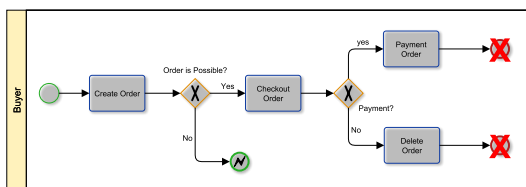
$NELE$ is the number of End Link Events

$NEMuE$ is the number of End Multiple Events

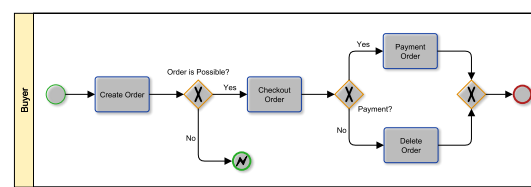
$NETE$ is the number of End Terminate Events

Convention on the modeling


Bad Modeling




Good Modeling



Restrict usage of terminate end event

Guideline Name	Guideline ID
Restrict usage of terminate end event	19
	 Terminate Event
Description	
<p>The process modeler should use terminate events only when strictly necessary; they are used to model situations where several alternative paths are enabled and the entire process have to be finished when one of them is completed. The modeler should use other end events rather than the terminate end event (e.g. a generic end event), to guarantee that the executions of the reaming process paths or activities will not be stoped.</p>	
Associated Metrics and Thresholds	
$MinimizeTerminateEndEvents(x) = \begin{cases} 0 & NETE = 0 \\ 1 & otherwise \end{cases}$ <p><i>where:</i> $x \in \text{Events} \wedge NETE$ is the number of End Terminate Events.</p>	

Explicit usage of gateways

Guideline Name	Guideline ID
Explicit usage of gateways	20
	 Gateway

Description

The modeler should not split or join flows using activities or events; the modeler should split or join sequence flows always using gateways. Moreover the modeler should not start conditional sequence flows from an activity; he should always make explicit use of gateways and start conditional sequence flows from them. This includes that an activity can have only one incoming sequence flow and only one outgoing sequence flow.

Source

[8, 19, 54, 55]

Associated Metrics and Thresholds

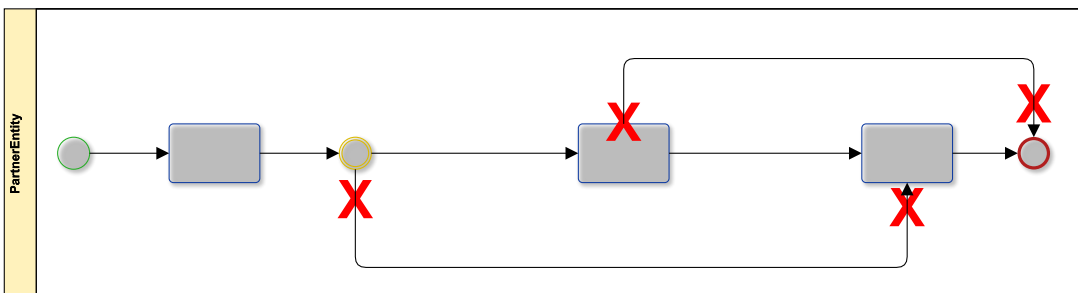
$$ExplicitGateway(x) = \begin{cases} 0 & \text{if (1)} \\ 1 & \text{otherwise} \end{cases}$$

where:

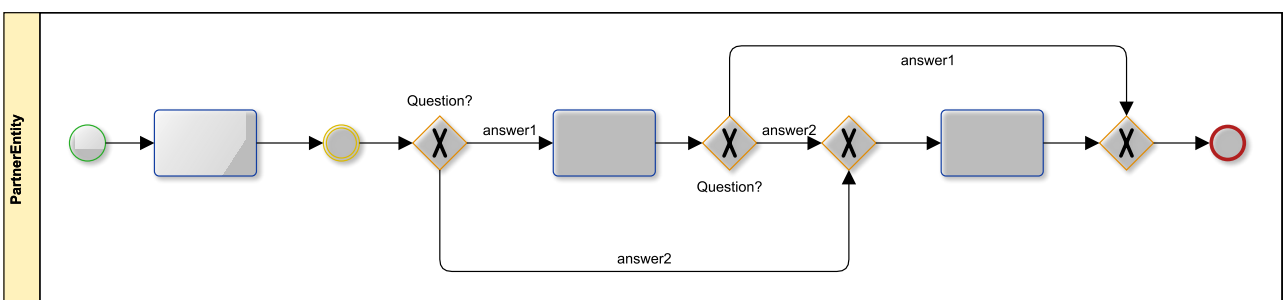
- (1) $(x \in Activities \cup IntermediateEvents \wedge Edges_{in}(x) = 1 \wedge Edges_{out}(x) = 1)$
 - $\vee (x \in StartEvents \wedge Edges_{in}(x) = 0 \wedge Edges_{out}(x) = 1) \vee (x \in EndEvents \wedge Edges_{in}(x) = 1 \wedge Edges_{out}(x) = 0)$
- Edges_{in}(x) is the sum incoming arc of element x Edges_{out}(x) is the sum outgoing arc of element x*

Convention on the modeling


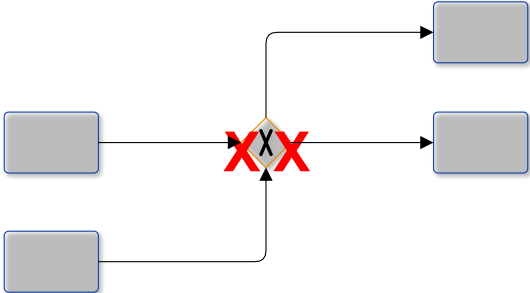
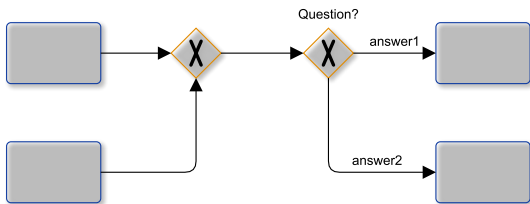
Bad Modeling




Good Modeling



Split and join flows

Guideline Name	Guideline ID
Split and join flows	21
	 Gateway
Description	
The modeler should not use gateways to join and split at the same time.	
Source	
[8, 19, 54, 55]	
Associated Metrics and Thresholds	
$SplitJoinFlow(x) = \begin{cases} 0 & \text{if } (Edges_{in}(x) = 1 \wedge Edges_{out}(x) > 1) \vee \\ & (Edges_{in}(x) > 1 \wedge Edges_{out}(x) = 1) \\ 1 & \text{otherwise} \end{cases}$	
<p>where:</p> <p>$x \in \text{Gateways}$.</p> <p>$Edges_{in}(x)$ is the sum incoming arc of element x</p> <p>$Edges_{out}(x)$ is the sum outgoing arc of element x</p>	
Convention on the modeling	
<p>Bad Modeling</p> 	<p>Good Modeling</p> 

Balance gateways

Guideline Name	Guideline ID
Balance gateways	22
	 Gateway

Description

The modeler should always use the same type of gateway used both for splitting and joining the flow. For example, when a flow is divided with a parallel gateway, the resulting parallel flows should be consolidated via another parallel gateway if or when required. In particular, the modeler should ensure that join parallel gateways have the correct number of incoming sequence flow-especially when used in conjunction with other gateways; this is related to ensuring the soundness property. Don't apply this guidelines on Event-based or Complex Gateways.

Source
 [54, 55]

Associated Metrics and Thresholds

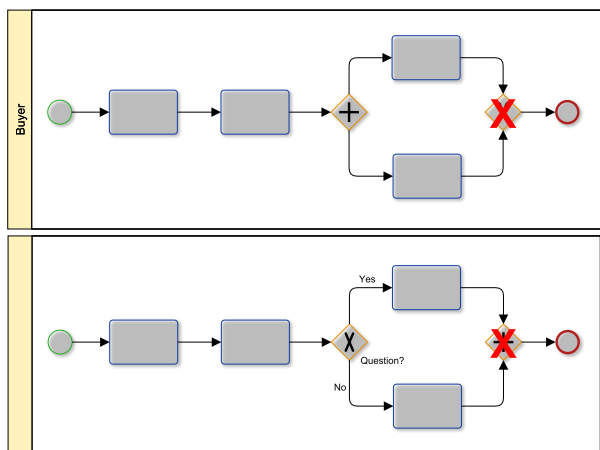
$$BalanceGateways(x) = \begin{cases} 0 & \text{if } GM \leq 15 \\ 1 & \text{otherwise} \end{cases}$$

where:

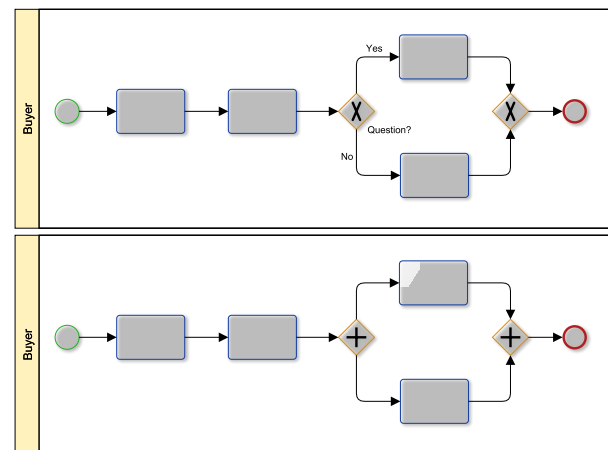
$x \in ExclusiveGateways \cup ParallelGateways \wedge GM$ is the Gateway Mismatch.

Convention on the modeling


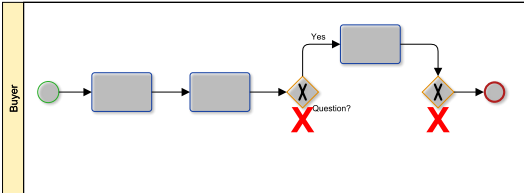
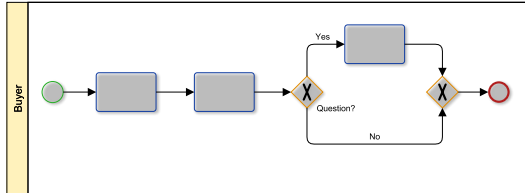
Bad Modeling




Good Modeling



Usage of meaningful gateways

Guideline Name	Guideline ID
Usage of meaningful gateways	23
	 Gateway
Description	
Since gateways are only used for linkage or merging within processes, they always need to have multiple incoming or outgoing flows. Gateways with only one incoming and one outgoing sequence flow do not provide any added value.	
Source	
[8, 19, 54, 55]	
Associated Metrics and Thresholds	
$MeaningfulGateways(x) = \begin{cases} 1 & \text{if } (Edges_{in}(x) = 1 \vee Edges_{out}(x) = 1) \\ 0 & \text{otherwise} \end{cases}$ <p>where: $x \in \text{Gateways}$</p>	
Convention on the modeling	
<p>Bad Modeling</p> 	<p>Good Modeling</p> 

Usage of inclusive OR-gateways

Guideline Name	Guideline ID
Usage of inclusive OR-gateways	24
	
	Inclusive Gateway
Description	
The modeler should minimize the use of inclusive OR-joins and inclusive OR-splits. Inclusive OR-splits activate one, several, or all subsequent branches based on conditions. They need to be synchronized with inclusive OR-join elements, which are difficult to implement in the general case.	
Source	
[54]	
Associated Metrics and Thresholds	
$\text{MinimizeOrGateway}(x) = \begin{cases} 0 & \text{if } NID = 0 \\ 1 & \text{otherwise} \end{cases}$ <p>where: $x \in \text{Gateways} \wedge NID$ is the number of inclusive decision</p>	

Usage of default flows

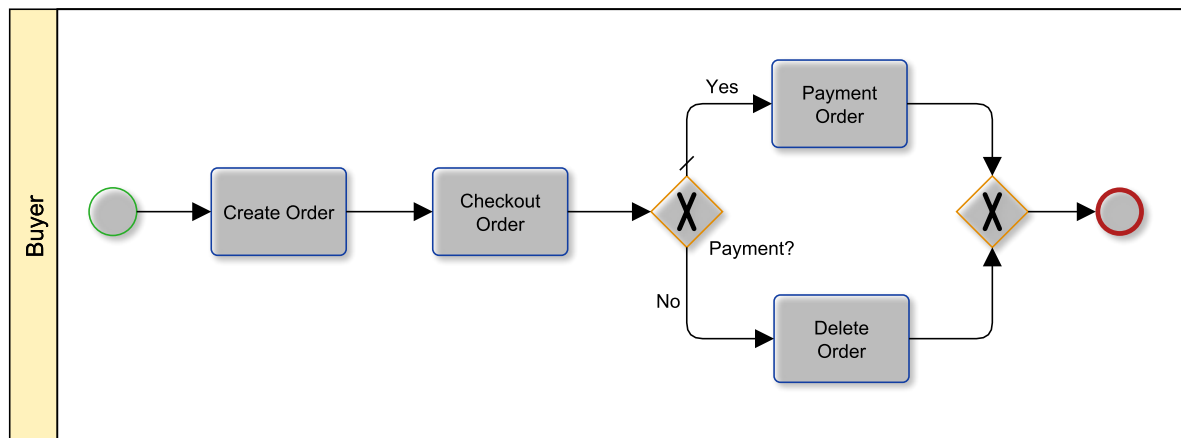
Guideline Name	Guideline ID
Usage of default flows	25
	→

Description


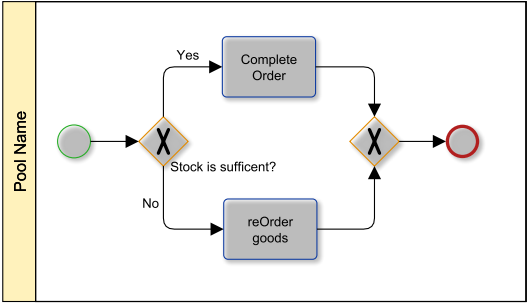
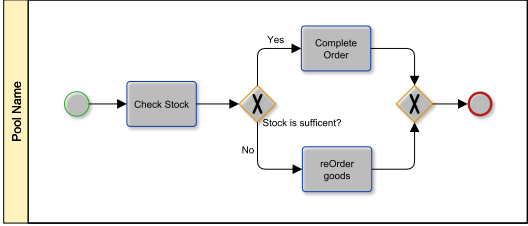
Where possible, after an exclusive and an inclusive gateway, the modeler should express the default flow in such a way that the default path is chosen if all the other conditions turn out to be false. One way for the modeler to ensure that the process does not get stuck at a gateway is to use a default condition for one of the outgoing sequence flow. This default sequence flow will always evaluate to true if all the other sequence flow conditions turn out to be false.

Convention on the modeling

Good Modeling



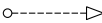
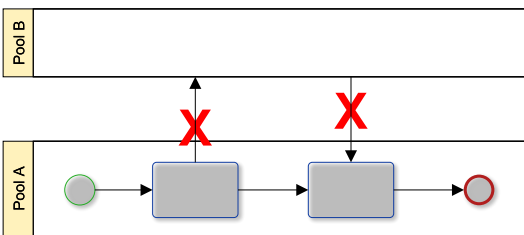
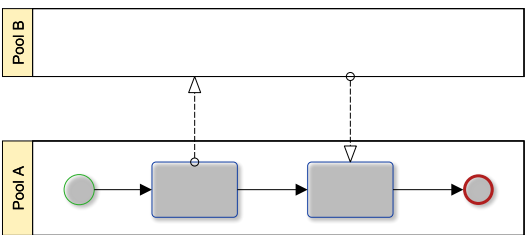
Usage of activities before splits

Guideline Name	Guideline ID
Usage of activities before splits	26
	
Description	
<p>For a comprehensive modeling, the modeler should always place an activity that determines the diverging condition(s) just before a diverging gateway of type exclusive, inclusive and complex.</p>	
Source	
<p>[19]</p>	
Convention on the modeling	
<p>Bad Modeling</p> 	<p>Good Modeling</p> 

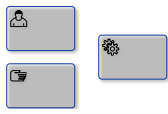
Message exchange

Guideline Name	Guideline ID
Message exchanges	27
Description	
<p>The modeler could represent message exchange with different elements. A clearer usage of those elements would be:</p> <ul style="list-style-type: none">• Send Task, can be used to express that the sending of a message requires an effort such as: making a phone call, sending an email, delivering a document, accessing a data store to retrieve data, etc.• Receive Task, can be used to express that the receiving of a message requires an effort such as: answering a phone call, checking the email, collecting documents, storing data on a data store, etc.• Intermediate Throwing Event, can be used to express that the sending of a message doesn't require particular effort e.g. the message is automatically processed by a system.• Intermediate Catching Event, can be used to express that the receiving of a message doesn't require particular effort e.g. the message is received and automatically processed by a system.• For other cases of message exchange, the modeler should use the remaining Message events such as: Message Start Event (if the process starts after receiving a message); Message Event Sub-Process Interrupting/Non-interrupting (if a received message starts a subprocess); Message Boundary Interrupting/Non-interrupting (if a message is received by a subprocess); Message End Event (if the process or subprocess, ends after sending a message).	

Usage of message flows

Guideline Name	Guideline ID
Usage of message flows	28
	
Description	
<p>For distinct and comprehensive modeling, message flows should only be used by the designated elements. Therefore, the difference between transmitter and receiver has to be considered. Moreover, the message flows should be used consistently. That is, if a particular message flow is shown in a subprocess nested three levels down, it should also be shown in the top-level process, and labeled the same way at every level. The modeler should represent message flows with all message events.</p>	
Convention on the modeling	
<p>Bad Modeling</p> 	<p>Good Modeling</p> 

Task types specification

Guideline Name	Guideline ID
Task types specification	29
	
Description	
The modeler should distinguish task types, e.g. manual task, user tasks and service tasks.	
Source	
[83]	

4.1.4. Labeling Guidelines


Document minor details

Guideline Name	Guideline ID
Document minor details	30
Convention concerning the name	
The model should contain only essential information therefore the modeler should leave details to documentation keeping labels simple and limiting the use of text annotations.	


Labeling convention

Guideline Name	Guideline ID
Labeling convention	31
Convention concerning the name	
Labels associated to process elements must be properly defined. The modeler should not use short names or abbreviations. The modeler should always use keywords that are meaningful to the business; he should not use the element type in its name. The name should emphasize the goal, and details of activity can be captured in comments or documentation. The modeler should not use conjunctions in names raise name abstraction level or split into two subsequent/alternative activities.	

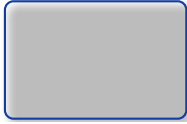
Labeling Pools

Guideline Name	Guideline ID
Labeling Pools	32
	
Convention concerning the name	
<p>The modeler should label pools using the participant's name. The main pool can be labeled using the process name. A Participant represents a specific PartnerEntity (e.g., a company) and/or a more general PartnerRole (e.g., a buyer, seller, or manufacturer) that are Participants in a Collaboration. A Participant is often responsible for the execution of the Process enclosed in a Pool. If a pool is present in a subprocess, the name of the pool must be the same of the superprocess pool which includes the subprocess activity. This means that the pool of the superprocess and the pool of the subprocess needs to be the same.</p>	
Source	
[19] http://www.bpmnquickguide.com/quickguide/rules/naming_conventions_best_practices.htm	


Labeling Lanes

Guideline Name	Guideline ID
Labeling Lanes	33
	
Convention concerning the name	
<p>Roles must always have a name. The naming of the used role should represent the responsible entity for the process. Lanes are often used to categorize elements by roles. They are often used for such things as internal roles (e.g., manager, associate), systems (e.g., an enterprise application), or internal departments (e.g., shipping, finance).</p>	
Source	
[19] http://www.bpmnquickguide.com/quickguide/rules/naming_conventions_best_practices.htm	



Labeling Activities

Guideline Name	Guideline ID
Labeling Activities	34
	
Convention concerning the name	
Activities labels should be composed of one verb, and one object. The verb used should use the present tense and be familiar to the organization. The noun has to be qualified and also of meaning to the business. Moreover, Multiple activities should not be labeled with the same name, except for same <i>Call Activities</i> used many time in the process.	
Source	
[83, 19, 54, 55]	


Labeling Events

Guideline Name	Guideline ID
Labeling Events	35
	
Convention concerning the name	
All events should have a label representing the state of the process: <ul style="list-style-type: none">• Events of type message, signal, escalation, and error events should be labeled with a past participle using an active verb;• Link events should be labeled with a noun;• Timer events should be labeled with time-date or schedule;• Conditional events should be labeled with the condition that triggers them.	
Source	
[19] http://www.bpmnquickguide.com/quickguide/rules/naming_conventions_best_practices.htm	


Labeling Start and End Events

Guideline Name	Guideline ID
Labeling Start and End Events	36
	  Start Event End Event
Convention concerning the name	
The modeler should not label start none and end none event if there is only one instance of them. The modeler should use labeling when multiple start and end events are used. Label them according to what they represent using a noun. Do not repeat names.	
Source	
[19] http://www.bpmnquickguide.com/quickguide/rules/naming_conventions_best_practices.htm	

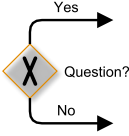
Labeling Message Event

Guideline Name	Guideline ID
Labeling Message Event	37
	
Convention concerning the name	
Whenever the modeler uses a message event he should draw the message flow, and label both the event and the message flow. The event should be labeled with the action Request X, for example and the message flow should be labeled with the name of the message.	
Source	
[19] http://www.bpmnquickguide.com/quickguide/rules/naming_conventions_best_practices.htm	


Labeling XOR Gateway

Guideline Name	Guideline ID
Labeling XOR Gateway	38
	
Convention concerning the name	
Diverging exclusive gateways should be labeled with an interrogative phrase. The name should be composed of one verb, one object, and a question mark to identify what is being evaluated.	
Source	
[19] http://www.bpmnquickguide.com/quickguide/rules/naming_conventions_best_practices.htm	

Labeling Gateway Outgoing Sequence Flows

Guideline Name	Guideline ID
Labeling Gateway Outgoing Sequence Flows	39
	
Convention concerning the name	
Sequence flows coming out of diverging gateways of type exclusive, inclusive and complex should be labeled using their associated conditions stated as outcomes.	
Source	
[19] http://www.bpmnquickguide.com/quickguide/rules/naming_conventions_best_practices.htm	


Labeling AND-Gateways

Guideline Name	Guideline ID
Labeling AND-Gateways	40
	
Convention concerning the name	
Labels on AND-splits and joins (and sequence flows connecting them) add no new information, so it is best to omit them. Since a sequence flow label describes a condition and AND-gateways are unconditional, the modeler should not label an AND-gateway or its gates.	
Source	
[19] http://www.bpmnquickguide.com/quickguide/rules/naming_conventions_best_practices.htm	

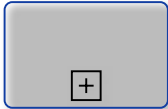
Labeling converging Gateways

Guideline Name	Guideline ID
Labeling converging Gateways	41
Convention concerning the name	
Converging gateways do not required to be labeled. When the convergence logic is not obvious, a text annotation should be associated to the gateway.	
Source	
[19] http://www.bpmnquickguide.com/quickguide/rules/naming_conventions_best_practices.htm	

Labeling Data Object

Guideline Name	Guideline ID
Labeling Data Object	42
	
Convention concerning the name	
All data objects should be labeled using a qualified noun that is the name of a business object. Multiple instances of the same data object are labeled (which are really data object references) using a matching label followed by the applicable state in square brackets.	
Source	
[19] http://www.bpmnquickguide.com/quickguide/rules/naming_conventions_best_practices.htm	


Consistent labeling of Subprocess

Guideline Name	Guideline ID
Consistent labeling of Subprocess	43
	
Convention concerning the name	
The name of the collapsed subprocess should receive the same name as the diagram and the subprocess should have the same name as the activity in the superprocess.	
Source	
[19] http://www.bpmnquickguide.com/quickguide/rules/naming_conventions_best_practices.htm	

Labeling synchronized end/split

Guideline Name	Guideline ID
Labeling synchronized end/split	44
Convention concerning the name	
It makes very straightforward to consistently use gateways and sub processes. Matching the label of a subprocess end state with the label of a gateway immediately following the subprocess allows to have a clear vision on how subprocess and process are linked together.	
Source	
[19] http://www.bpmnquickguide.com/quickguide/rules/naming_conventions_best_practices.htm	

Loop Marker Annotation

Guideline Name	Guideline ID
Loop Marker Annotation	45
	
Convention concerning the name	
The loop marker provides a more compact representation than the gateway-loopback, but it hides the loop condition. For that reason, it is best to indicate the condition in a text annotation. Note: a condition of the form Until X corresponds to the loop condition if Not X; when X is true, Not X is false and the looping ends.	

4.1.5. Patterns Guidelines

Apply process patterns

Guideline Name	Guideline ID
Apply process patterns	46
Description	
The modeler should use modeling patterns (BPMN workflow patterns see http://www.omg.org/bpmn/Documents/Notations_and_Workflow_Patterns.pdf) to model the required business conditions while simplifying the model. But he should use only the variations that comply with the guidelines.	

Reduce the number of redundant activities

Guideline Name	Guideline ID
Reduce the number of redundant activities	47
Description	
If a set of consecutive activities (without boundary events) can be performed by the same person, then these activities could be integrated into a single activity or could be represented in a subprocess. A set of consecutive activities in the same lane (or in a pool without lanes) may indicate missing participant details, too much detail, or a misalignment in scope.	

Use of subprocesses

Guideline Name	Guideline ID
Use of subprocesses	48
Description	
Use sub-processes to group activities with the same purpose when: 1) A set of consecutive activities has an owner different from the main process owner; 2) A set of consecutive activities has a different goal from the main process one; 3) A process or a fragment must be re-used in another process (use Call Activities in this case).	

Use subprocesses to scope attached events

Guideline Name	Guideline ID
Use subprocesses to scope attached events	49
Description	
Intermediate events attached to a process activity imply that if the event occurs while that activity is running, it aborts the activity and the process proceeds down the sequence flow out of the event. A subprocess with attached event enables to clearly define the scope of an event. If the response to the handling of an exception (in the use of boundary events) is the same for every activity within a contiguous segment of the process, the modeler should not attach the same boundary to each of those activities and merge the exception flows. The correct way to model it is to enclose that segment in a subprocess and attach a single boundary event to the subprocess boundary.	

Use a timer intermediate event with an event gateway

Guideline Name	Guideline ID
Use a timer intermediate event with an event gateway	50
Description	
One way for the modeler to ensure that the process does not get stuck at an event based exclusive gateway is to use a timer intermediate event as one of the options for the gateway.	

Service task

Guideline Name	Guideline ID
Service task	51
Description	
Use send and receive tasks for long-running services, service task for short-running. In an executable process, synchronous tasks are short-running, completing in milliseconds or seconds. If an automated task is long-running, meaning it takes minutes, hours, or even weeks to complete, it is modeled as an asynchronous request, using a send task or throwing message event, not a service task. If an automated function is long-running, represent it with separate send and receive tasks (with message flows). Reserve service task for short-running actions.	

4.1.6. Appearance Guidelines

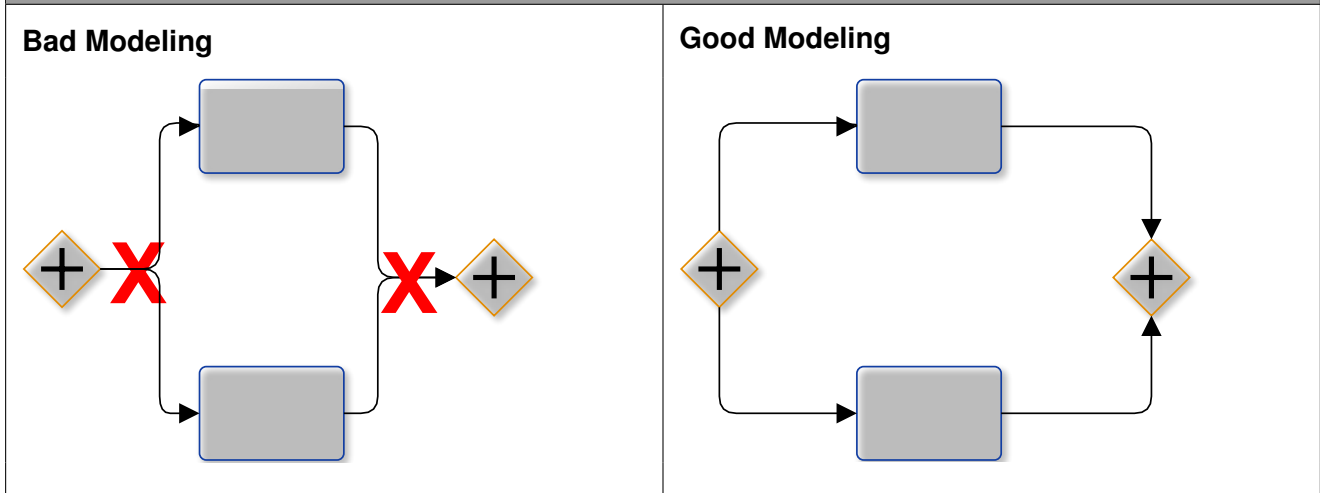
Neat and consistent model

Guideline Name	Guideline ID
Neat and consistent model	52
Description	
The modeler should keep the model as neat and consistently organized as possible.	

Absence of edge overlays

Guideline Name	Guideline ID
Absence of edge overlays	53
Description	
If edges are Sequence, Default, Condition or Message Flow the overlapping of edges is not allowed. The comprehension and the clarity of the model will suffer under the violation of the existing rule.	
Source	
[8, 19]	

Convention on the modeling



Absence of node intersections

Guideline Name	Guideline ID
Absence of node intersections	54
Description	
The nodes should not overlap other nodes. The comprehension and clarity of the model will suffer under the violation of this rule.	
Source	
[8, 19]	
Convention on the modeling	
<p>Bad Modeling</p>	<p>Good Modeling</p>

Avoid crossing flows

Guideline Name	Guideline ID
Avoid crossing flows	55
Description	
The modeler should try to avoid crossing flows as much as possible. This will increase the understanding of process models for both experienced and inexperienced users.	
Source	
[8, 19]	
Convention on the modeling	
<p>Bad Modeling</p>	<p>Good Modeling</p>

Consistent edge folding in sequence flows

Guideline Name	Guideline ID
Consistent edge folding in sequence flows	56
Description	
The consistent edge folding in sequence flows will be used to get a consistent layout in the workflow.	
Source	
[8, 19]	
Convention on the modeling	
<p>Bad Modeling</p>	<p>Good Modeling</p>

Consistent edge folding in message flows

Guideline Name	Guideline ID
Consistent edge folding in message flows	57
Description	
Message flows should be modeled constant without any foldings.	
Source	
[8, 19]	
Convention on the modeling	
<p>Bad Modeling</p>	<p>Good Modeling</p>

Placing messages between pools

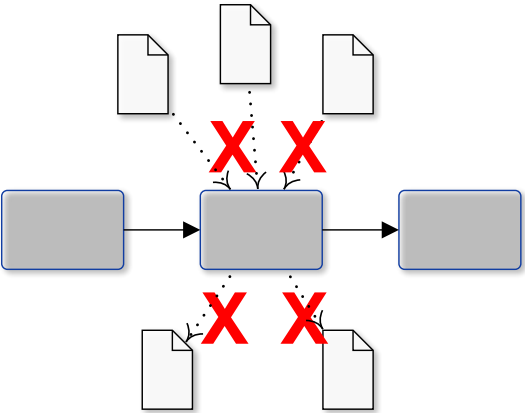
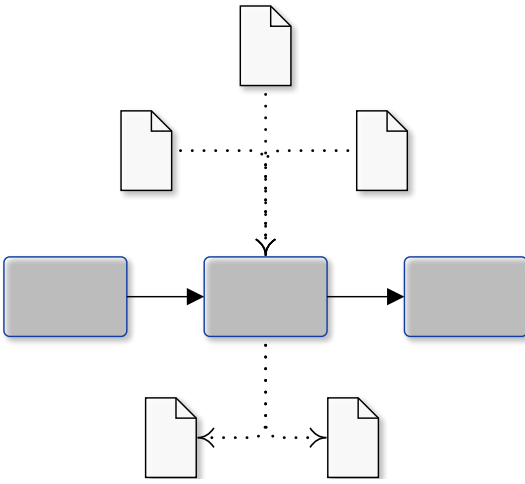
Guideline Name	Guideline ID
Placing messages between pools	58
Description	
If there is a communication between pools, this should be represented by messages. Therefore, the modeler should represent message flows with all message events.	
Source	
[8, 19]	

Convention on the modeling	
<p>Bad Modeling</p>	<p>Good Modeling</p>

Process orientation

Guideline Name	Guideline ID
Process orientation	59
Description	
The modeler should draw pools horizontally and use consistent layout with horizontal sequence flows, and vertical message flows and associations.	
Source	
[8, 19]	

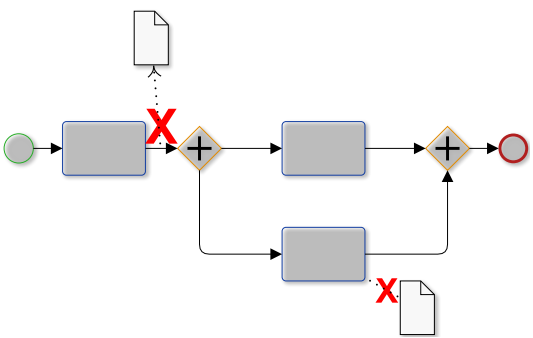
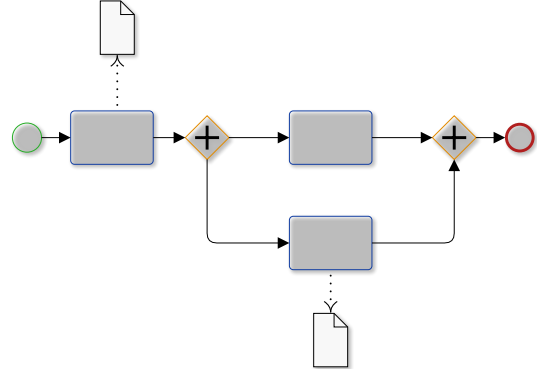
Group incoming/outcoming Artifacts

Guideline Name	Guideline ID
Group incoming/outcoming Artifacts	60
Description	
Inputsets are used to group incoming Artifacts. If there is more than one inputset, the modeler should pick a point on the boundary of an activity and have all inputs that belong to a single inputset connect to that point. The inputs for the other inputsets should each connect to separate points on the activity boundary. The same pattern should apply to modeling outputsets.	
Source	
[8, 19]	
Convention on the modeling	
Bad Modeling	Good Modeling
	

Keep a standard format

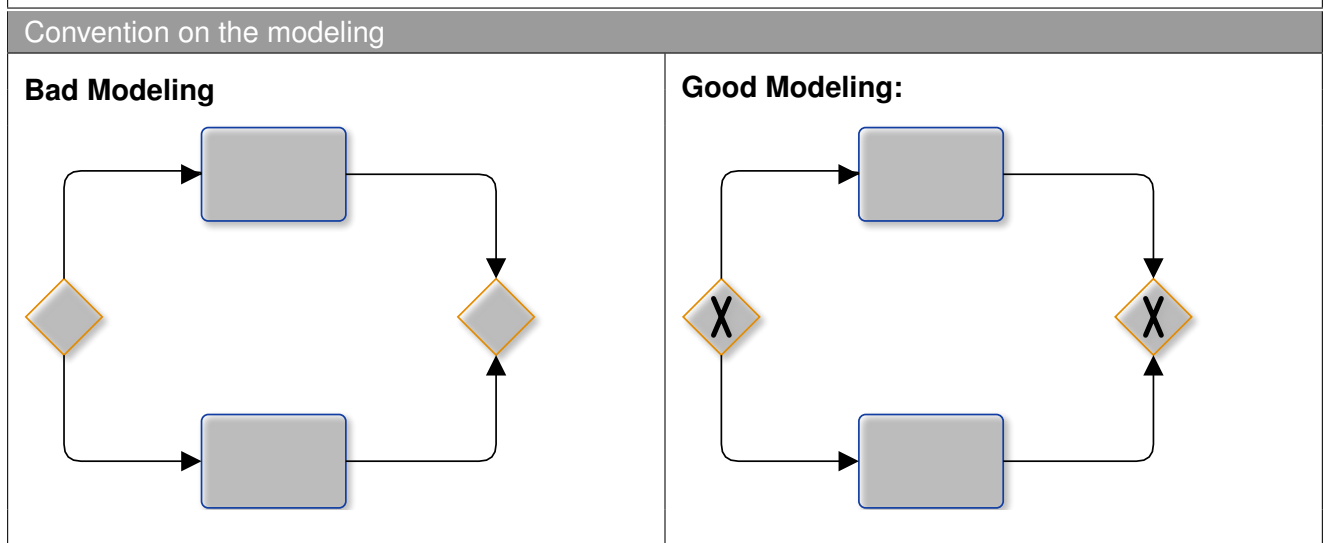
Guideline Name	Guideline ID
Keep a format standard	61
Description	
The modeler should keep a unique format along diagrams and focus on a clean and friendly look and feel. Using different font sizes, colors, boxes sizes or overlapping labels might make the diagrams reading a challenge. The specified colors should not be changed, in order to make diagrams recognisable. Further properties shall not be modeled with different colors.	
Source	
[8, 19]	

Data object association

Guideline Name	Guideline ID
Data object association	62
Description	
The modeler should associate data objects to activities. In particular the modeler should not associate a data object with a sequence flow if the sequence flow is connected to a gateway. The associations should always be modeled with the chosen edge direction.	
Convention on the modeling	
Bad Modeling	Good Modeling
	

Explicit Exclusive Gateway

Guideline Name	Guideline ID
Explicit Exclusive Gateway	63
Description	
The modeler should use the Exclusive Gateway with the marker "X" instead of using it without marker.	
Source	
[8, 19]	



4.2. Understandability into Practice

In this section we apply the guidelines presented in the previous section, in order to check the quality, with respect to the understandability, of the models specified in WP 8.1 for the Monti Azzurri Consortium. In particular, we consider the process model specified in WP8.1 Fig. 4.25 for the Monti Azzurri Consortium, (see Fig.4.1), and we redesign this model in order to make it more comprehensible, following the defined guidelines.

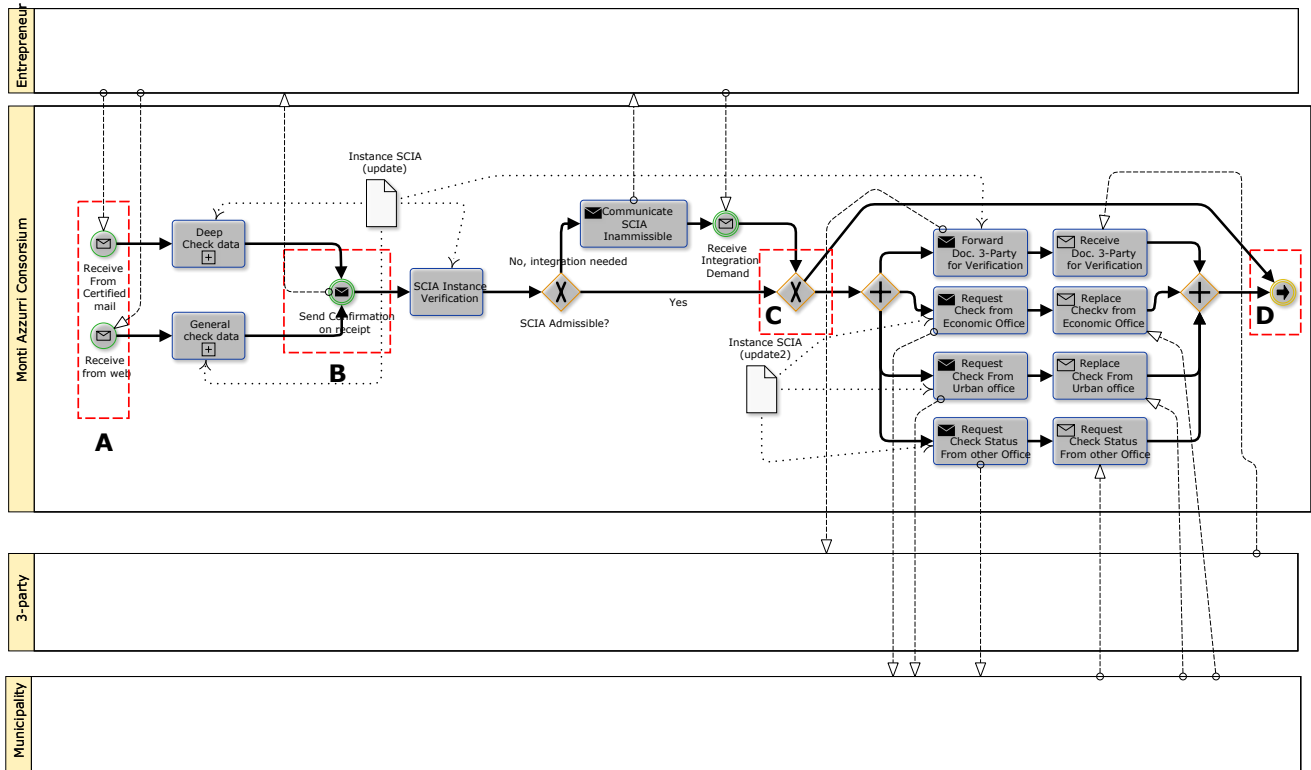


Figure 4.1: Monti Azzurri Consortium SCIA Commerciale BP of the Municipality Fig. 4.25 of WP8.1.

The model in Fig. 4.1 contains different cases of bad modeling, which make the overall model less understandable. In order to enhance the understandability of the model we apply the following guidelines:

- “ID 17 Consistent usage of start events” see dotted rectangle A of Fig. 4.1.
- “ID 20 Explicit usage of gateways” see dotted rectangle B and D of Fig. 4.1.
- “ID 21 Split and join flow” see dotted rectangle C of Fig. 4.1.
- “ID 57 Consistent edge folding in message flows” see all message flows of Fig. 4.1.

By applying these guidelines, we redesign the model of Fig. 4.1, as shown in Fig. 4.2. Now the model is more “linear”.

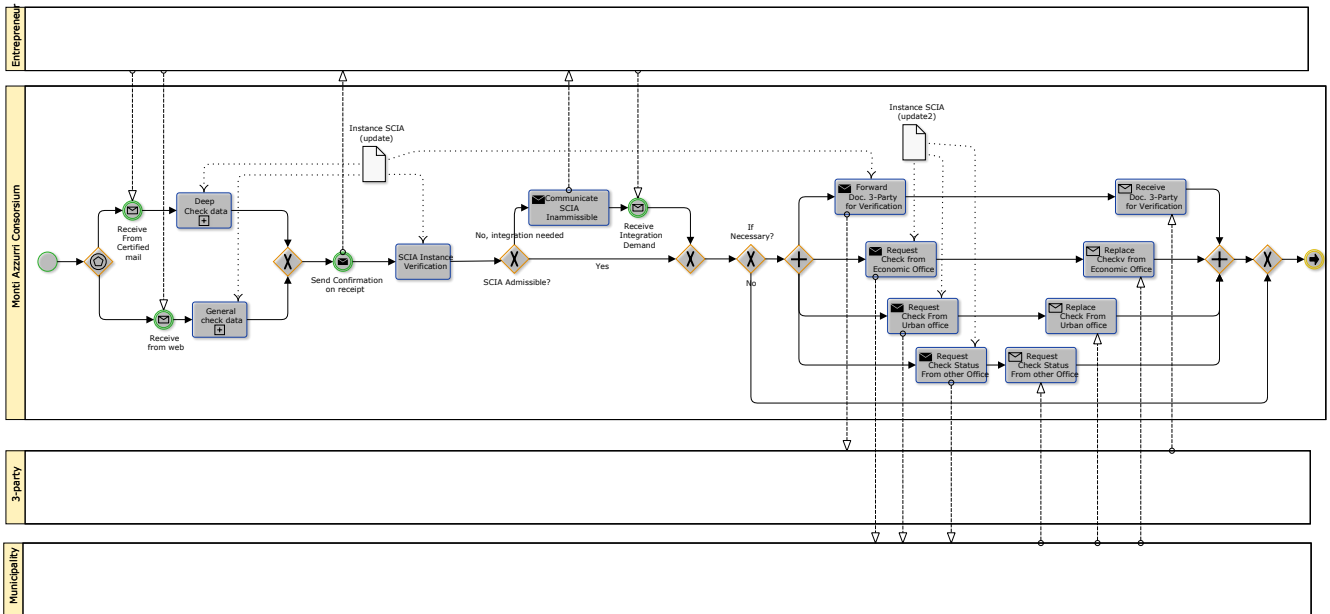
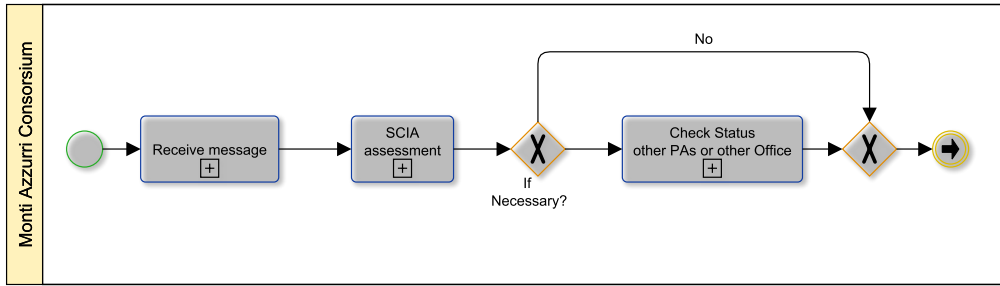
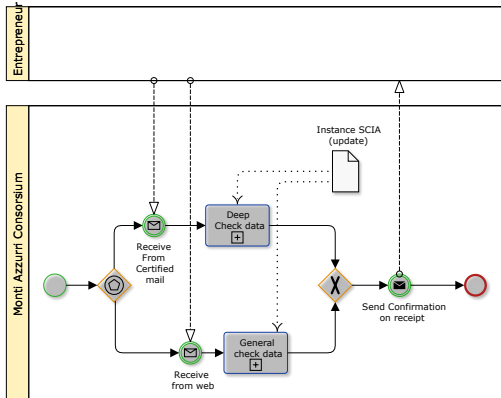


Figure 4.2: Monti Azzurri Consortium SCIA Commerciale BP of the Municipality

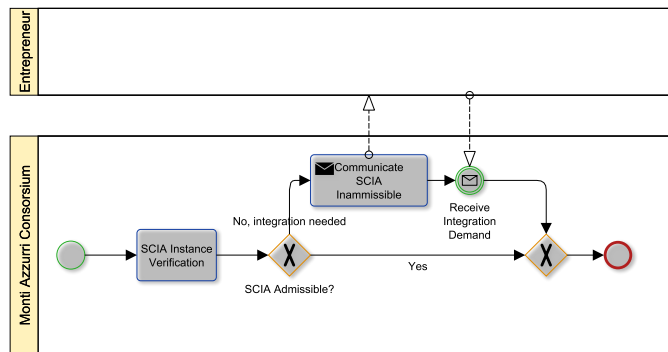
These are not the only applicable guidelines. If we wish to increase the understandability of the model, we should consider also the general guidelines, as “ID 5 Apply hierarchical structure with Sub-Processes” and “ID 7 Minimize model size”. We apply these two simple guidelines, as shown in Fig. 4.3. The process is divided into three subprocesses and this make the models much smaller and easier to understand.



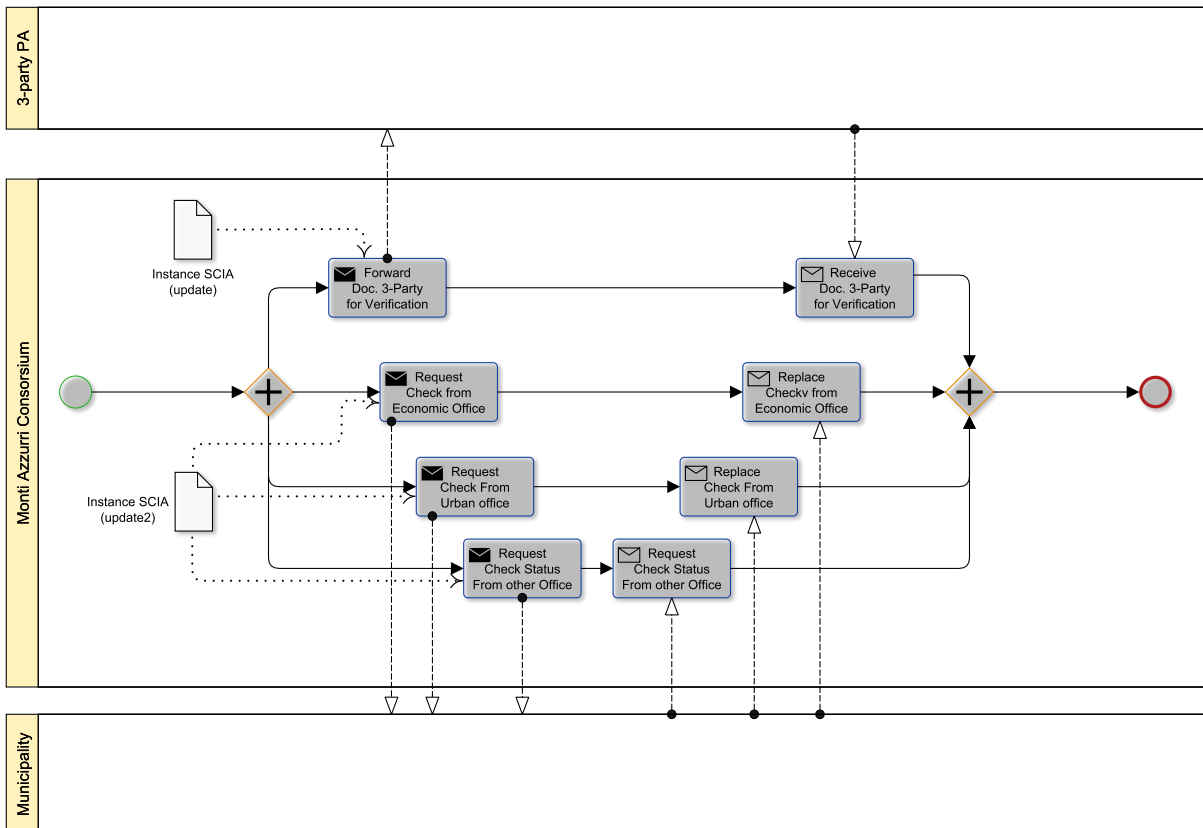
(a) "Monti Azzurri Consortium SCIA Commerciale"



(b) Subprocess Receive message



(c) Subprocess SCIA Assessment



(d) Subprocess Check Status other PAs or other Office

Figure 4.3: Monti Azzurri Consortium "SCIA Commerciale" BP of the Municipality.

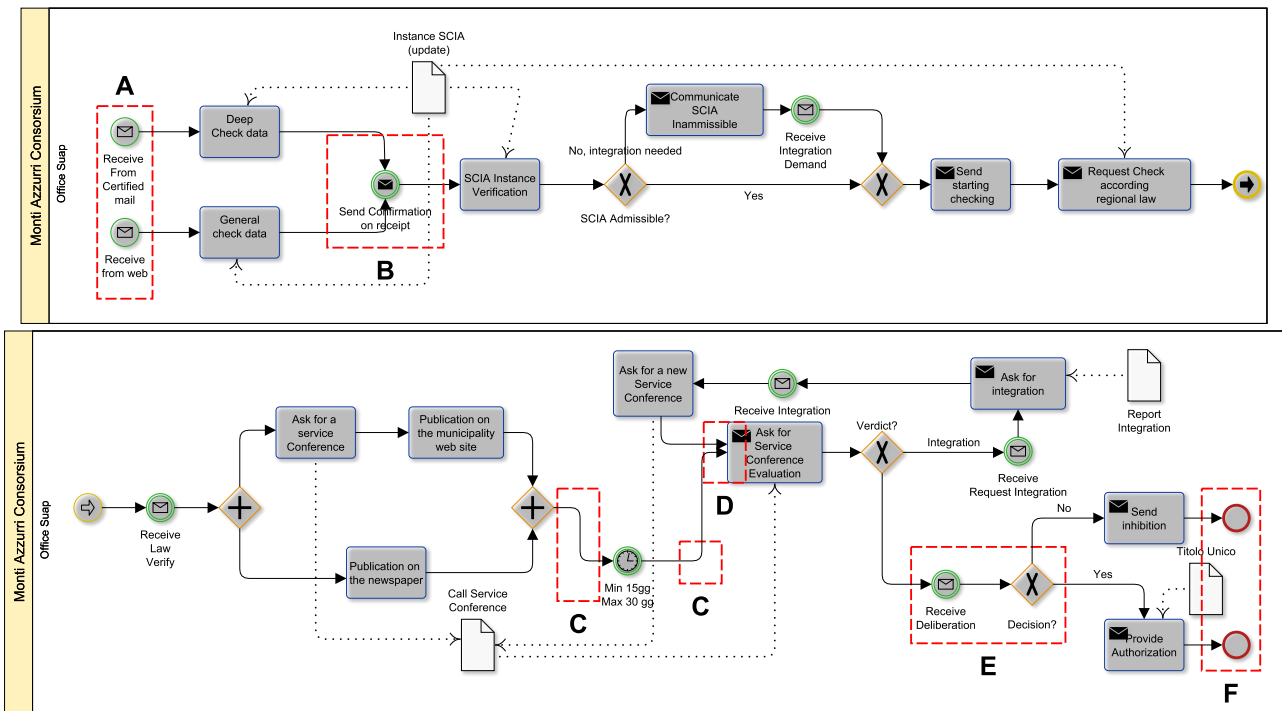
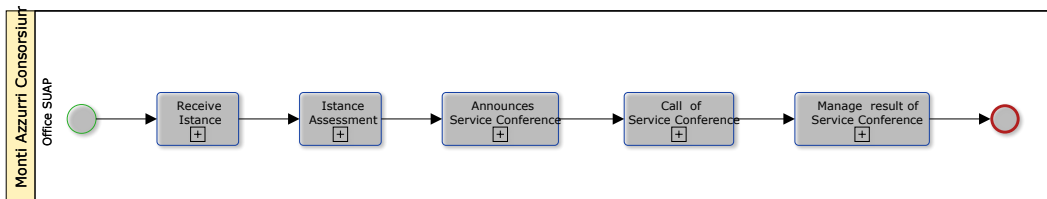


Figure 4.4: Monti Azzurri Consortium “Variante Urbanistica” BP Fig. 4.39 of WP8.1.

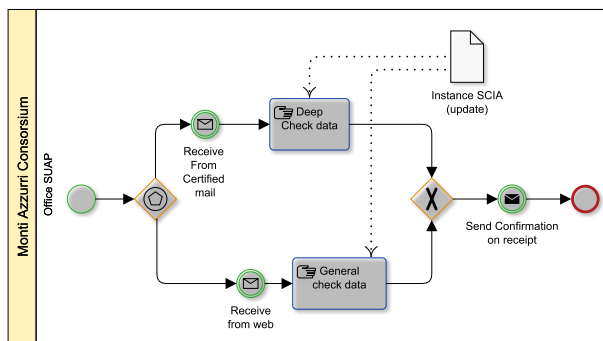
Another example of use of the guidelines is visible in Fig. 4.5, whereas the original model is visible in the Fig. 4.4 extracted from WP8.1. In order to enhance the understandability of the model the following guidelines were applied:

- “ID 5 Apply hierarchical structure with SubProcesses”.
- “ID 7 Minimize model size”.
- “ID 17 Consistent usage of start events” see dotted rectangle A of Fig. 4.4.
- “ID 20 Explicit usage of gateways” see dotted rectangle D of Fig. 4.4.
- “ID 26 Usage of activities before splits” see dotted rectangle E of Fig. 4.4.
- “ID 18 Consistent usage of end events” see dotted rectangle F of Fig. 4.4.
- “ID 56 Consistent edge folding in sequence flows” see dotted rectangle C of Fig. 4.4.

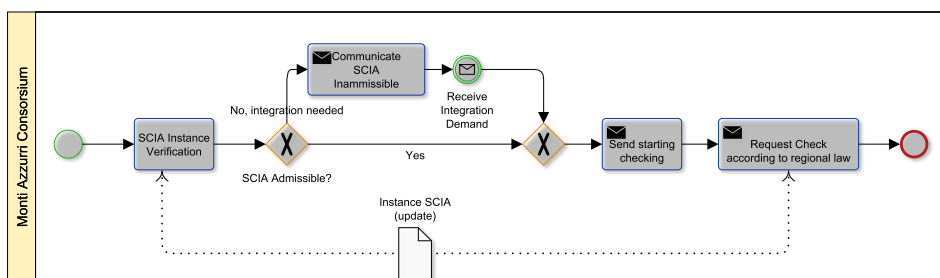
Now the model is smaller and easier to understood.



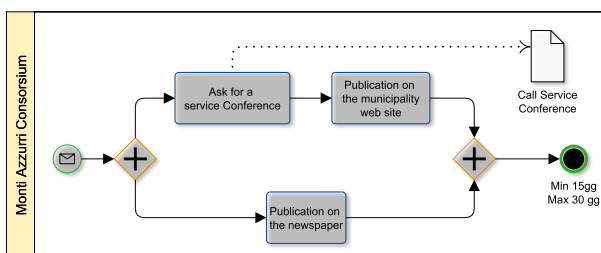
(a) "Monti Azzurri Consortium Variante Urbanistica"



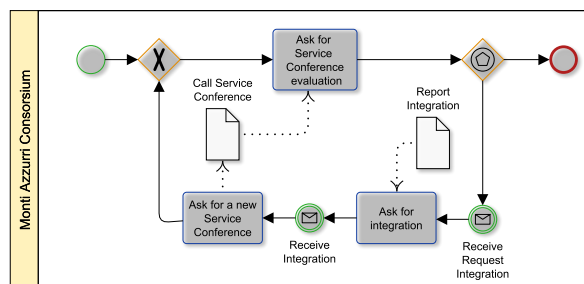
(b) Subprocess Receive message



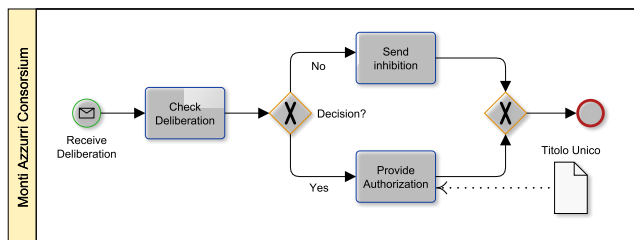
(c) Subprocess SCIA Assessment



(d) Subprocess Announces Service Conference



(e) Subprocess Call of Service Conference



(f) Subprocess Manage result of Service Conference

Figure 4.5: Monti Azzurri Consortium "Variante Urbanistica" BP of the Municipality.

5 Business Process Correctness

In this chapter we introduce BP formal verification techniques. Generally, correct and robust execution of a process model is fundamental. Consequently, the verification of BP models constitutes a fundamental task during process implementation. In particular, at build-time it has to be ensured that process models being deployed can be properly executed; corresponding BP instances will always complete in a well-defined and proper state. In particular, a BP execution must not be blocked due to modeling errors (e.g., non-satisfiable execution constraints) leading to deadlocks or other kinds of severe problems (for example missing initialization of data objects). By verifying BP models at build-time, potential problems can be identified and the model can be corrected before its deployment.

In this chapter we focus on the mapping from BPMN to Petri Net, on the verification of correctness for a BP model through the unfolding technique, and we introduce some further development that may be introduced to the Learn PAd platform.

5.1. Business Process Formalization

BPMN 2.0 is considered by domain experts as the most practical language in order to combine detailed and high-level system design. Practitioners can easily use it in order to model complex and inter-organizational BPs, nevertheless its main drawback is that it lacks of a formal semantic. The standard is defined in narrative form and in some case it could lead to misunderstanding and possibly wrong interpretations. In order to enable the use of formal analysis tools BPMN needs to be mapped into languages supported by a precise semantics. Among those languages, PN results to be a suitable tool for modeling and analyzing of concurrent systems. At the same time there are many tools that are ready to use for PN. Finally, PN graphical notations can be suitable to report the result of analysis in a user-friendly manner.

In the literature several general-purpose mappings from BPMN to PN are available [33], [44], [66]. Nevertheless most of them just consider a small subset of the whole specification. The following are probably the most interesting contributions in the area of mapping BPMN to PN.

- In [65] the authors present a tool for converting BPMN to PNs extended with inhibitor and reset arcs. Such models are then converted to the process specification language $\mu CRL2$ and verified against a wide variety of formally defined properties. Unfortunately the proposed mapping is not comprehensive enough in order to clearly address all the relevant BPMN elements typically used in a e-government modeling setting, for instance messages, sub-process and loop activity are not included.
- In [3] the authors provide a formalization of basic data object processing in BPMN putting them together with the more traditional control flow. Data are considered as precondition and effect of control flow. The proposed solution is part of a comprehensive work that concentrate on data flow, nevertheless it misses a general mapping of basic BPMN elements such as sub-processes and messages.
- To our knowledge the most complete mapping to transform BPMN models into PNs is that presented by Dijkman et al. [18]. Basic mapping rules are shown in 5.1. A task is mapped into a

transition with one input place and one output place. A start or end event is mapped onto a similar module except that a silent transition is used to signal when the process starts or ends. Gateways, except event-based decision gateways, are mapped into small PN modules with silent transitions capturing their routing behaviour. Starting from the basic elements the mapping also introduces some detail with respect to the concurrent execution of multiple instances, sub-processes and exception handling. We base our work on the Dijkman approach since it is quite complete, in comparison to the element that we use in Learn PAd and it can be easily extended to include the whole set of elements typically used in the specific domain. Moreover there is a previous experience in the use of the mapping for BP analysis in the e-government domain [9].

BPMN Object	Petri-net Module	BPMN Object	Petri-net Module
Start s		End e	
Message E		Task T	
Fork F1		Join J1	
(Data-based) Decision D1		Merge M1	

Figure 5.1: Selected mapping from BPMN to Petri Net (Dijkman et al. [18]).

A mapping rule, is also available for the BPMN element “subprocess”. A subprocess can be seen as a standalone process. In Fig. 5.2 we can see a mapping restricted to subprocesses with a single start event and a single end event only, and without exception handling.

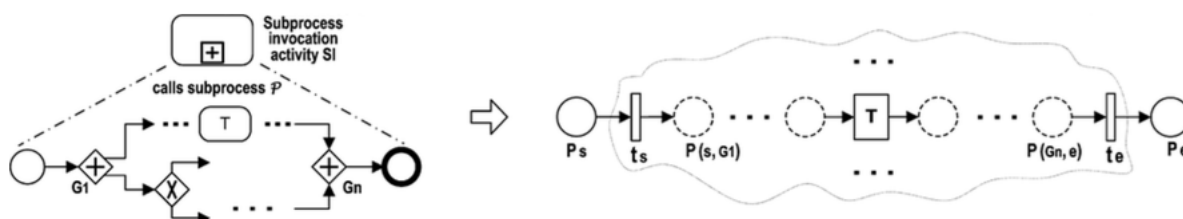


Figure 5.2: Mapping of a sub-process without exception handling (Dijkman et al. [18]).

A message flow describing the interaction between processes, can be mapped to PN. In [18], the authors distinguish four kinds of mapping, based on the type of elements sending and receiving the message. In addition, a task may be replaced by an intermediate message event without changing the rule. The mapping in 5.3 is restricted to tasks that either send or receive messages but this doesn't limit the expressive power of BPMN, because sending and receiving a message can be represented by two tasks: one *send* and one *receive*.

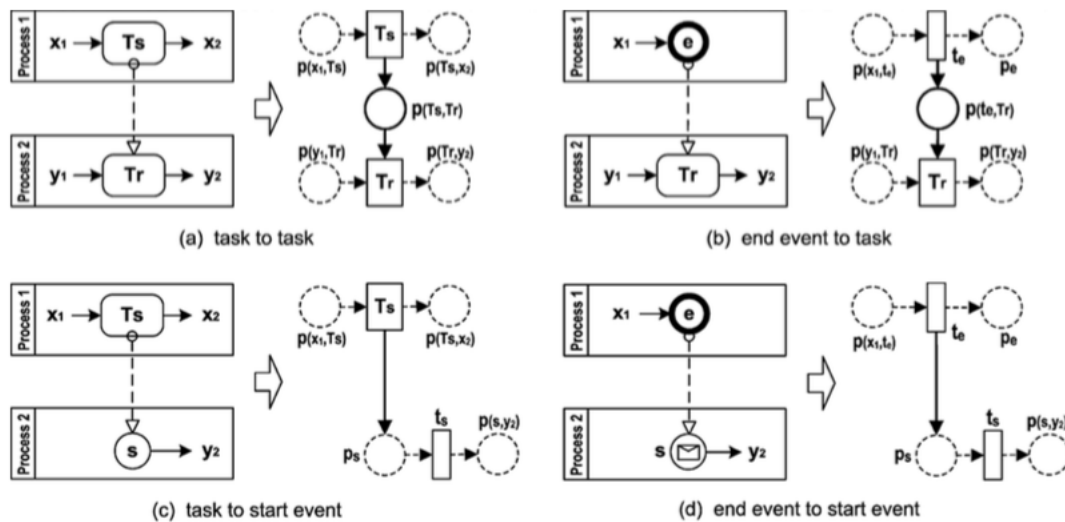


Figure 5.3: Mapping of message flows between BPMN processes. (Dijkman et al. [18]).

In order to better understand how the mapping works, Fig. 5.4, Fig. 5.5, Fig. 5.6, and Fig. 5.7 report the mapping of each BPMN element used to represent the behavior of Participant B. A complete view on the selected mapping is shown in Fig. 5.7, it refers to the transformation from BPMN Business Process (A) to a Petri Net (B). In the basic example the BP includes two participants (pools) exchanging a message. Both participants internally decide on how to behave according to the assessment of the choice statement. It is possible to observe the resulting model for the message sent by participant A to participant B represented by the place p_{20} in the PN.

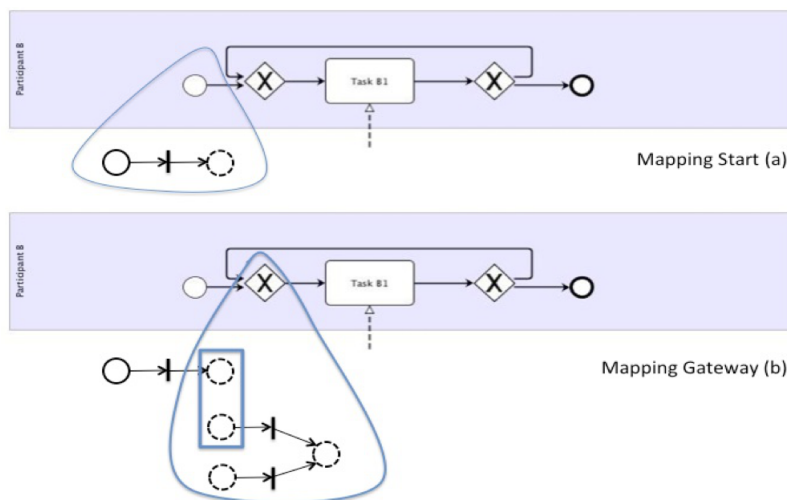


Figure 5.4: Mapping Participant B: start event and gateway.

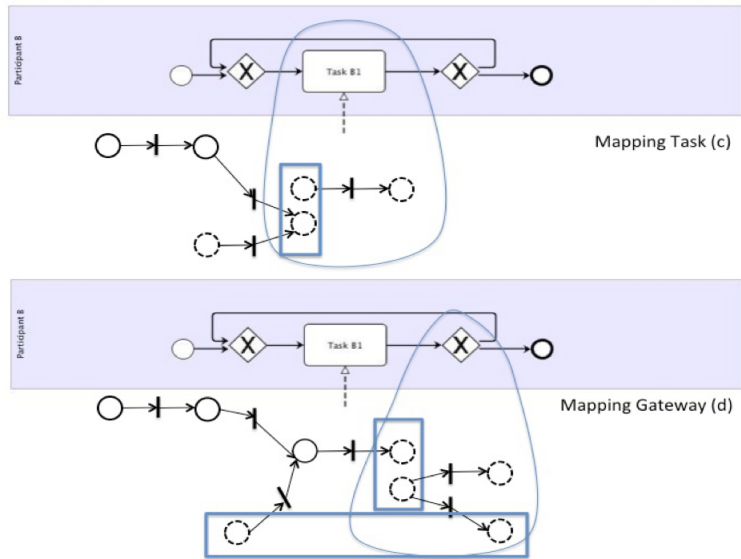


Figure 5.5: Mapping Participant B: task and gateway.

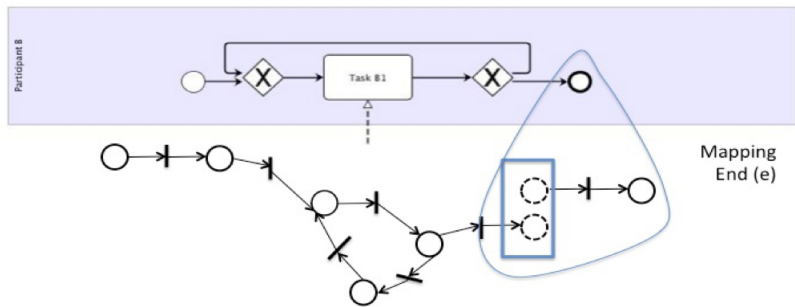


Figure 5.6: Mapping Participant B: end event.

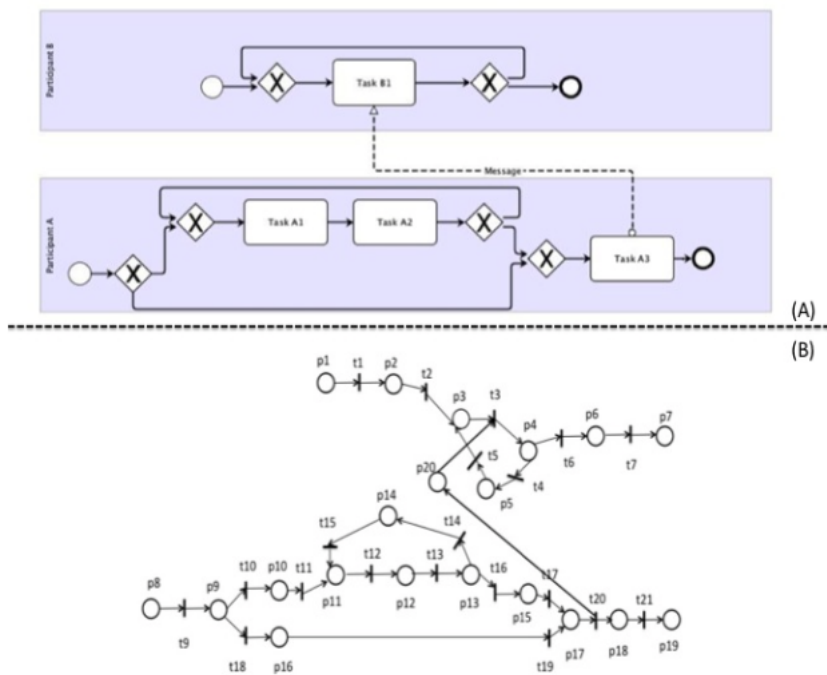


Figure 5.7: Example of the resulting PN for an intra-organizational BP.

5.2. Correctness Verification

Considering correctness verification and the mapping from BPMN to Petri Net provided by Dijkman in [18] it is possible to verify several BP model properties. In the Learn PAd project we focus on those that are related to the control flow correctness. Verification techniques are also investigated and in order to reduce the state explosion problem we consider unfolding as a valid analysis solution.

5.2.1. Control Flow Correctness

The properties we are able to verify with respect to the control flow correctness are reported and described in the following. All of them will be supported by the model verification component.

A **Reachability** property states that a particular situation can sometimes be reached [43]. For any BP model it is of particular interest to verify the reachability of the goal state(s) and to examine the sequences of activities through which the goal state(s) can be reached.

A **Liveness** property expresses that, under certain conditions, a situation will ultimately occur. This formulates a much stronger condition than reachability [43]. A liveness property requires that, independent of the system behavior, a particular situation can always be reached. Termination of a process is a liveness property. Whereas reachability only requires that there is a trace of behavior leading to the goal state, liveness requires that the goal state is reachable via any of the possible traces of behavior.

The **Soundness** property indicates that a process with a start marking of k tokens in its sink place, can reach the termination state marking with k tokens in its sink place. Additionally, all the transitions in the process could fire (i.e., for each transition there is a reachable state in which the transition is enabled). This property is verified when three requirements are satisfied [67]:

- *Option to complete.* A process instance, once started, can always complete.
- *Proper completion.* When a process instance completes, there exists no related activity of this instance which is still running or enabled.
- *No dead activities (or Deadness).* A process model does not contain any dead activity, i.e., for each activity there exists at least one completed trace producible on that model and containing this activity.

We believe that, it is easier to understand those properties with counterexamples, then we show some of them to illustrate the failure of satisfying the soundness requirements.

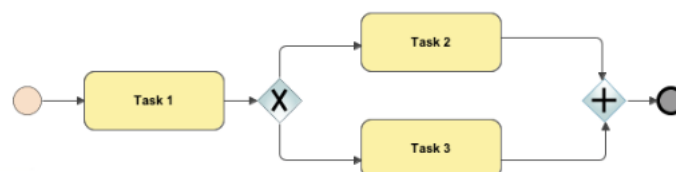


Figure 5.8: BPMN example of the failure to satisfy Option to complete.

In Fig. 5.8 we can see an example of failing to satisfy the option to complete requirement. The BP in fact is stuck at the parallel gateway waiting for both the incoming sequence flows to be activated (after the execution of Task 2 and Task 3). However, the initial Xor gateway prevents the activation of both sequence flows then the process will never execute Task 2 and Task 3 (the process will execute only one of them) which are instead necessary to complete the process.

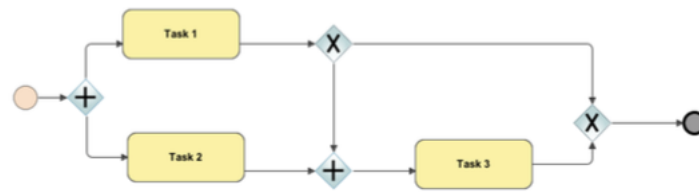


Figure 5.9: BPMN example of the failure to satisfy Proper Completion.

In Fig. 5.9 we can see an example of failing to satisfy the Proper Completion requirement. The BP can actually complete immediately after the execution of Task 1 without waiting for the execution of Task 2 or Task 3 which may be still running.

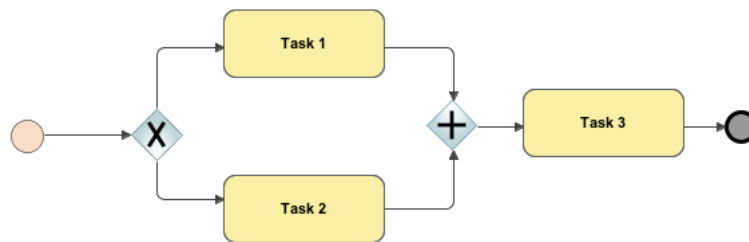


Figure 5.10: BPMN example of the failure to satisfy No dead activities.

In Fig. 5.10 we can see an example of failing to satisfy the No dead activities requirement. Task 3 is considered a Dead activity since it will never be executed. The parallel gateway requires both Task 1 and Task 2 to be executed while the Xor gateway allows the execution of only one of them, therefore the parallel gateway will never activate and Task 3 will never be executed.

The notion of Soundness we presented above, is also called *Classical Soundness*; this is for distinguish it from other form of Soundness such as: Lazy Soundness, Easy Soundness, Weak Soundness, etc. Those forms, relax one or more requirement of Classical Soundness; a detailed explanation can be found in [87].

- **Weak Soundness.** The weak soundness property relaxes the option to complete criterion, to say that, it is possible to complete a process in some cases, when started (Weak option to complete).
- **Relaxed Soundness.** This notion allows for potential deadlocks and livelocks, however, for each transition there should be at least one proper execution.
- **Lazy Soundness.** It weakens the requirement of ending in a state with no tokens in any place other than the sink place, i.e., tokens may be left behind as long as the sink place is marked precisely once. This property is verified only if option to complete and proper completion are satisfied.
- **Easy Soundness.** It only considers the possibility of the option to complete.

5.2.2. Unfolding

The problem of state space explosion for the verification of BP models, and in particular for Petri Nets, is well-known. To challenge this problem, we make use of the unfolding method since it explore the state space of concurrent systems without considering all possible events interleaving. This makes possible to analyze very big systems without exhaustively enumerating all the behaviors or reachable states. Unfolding is generally applied to Petri Nets [51] even if application to general transition systems has also been proposed in the literature [21] (see Appendix C for details). The unfolding of a Petri Net

can be described as an infinite net of a restricted form called Occurrence Net. It allows us to choose a partial rather than a total order on moves, and thereby avoid unnecessary bifurcation in the search. By applying the McMillan unfolding algorithm [51] to the Petri Net resulting from the BPMN 2.0 mapping it is possible to observe if properties, such as deadlocks freedom, are satisfied or not by a BP specification. The basic version of the unfolding provided by McMillan was selected due to the characteristics of BPs in e-government where it seems reasonable to have BPs resulting in bounded and safe Petri Net models after the mapping.

At this point it is important to underline the relation between unfolding and firing sequences introducing the notion of configuration. A configuration represents a possible partial run of the net. This is any set of transitions that satisfies the following conditions: (i) If we take a transition in the configuration and we race a path backward along the arrows all the transitions we encounter must also be in the configuration; (ii) Two transitions in the configuration cannot be in conflict, meaning that both of them should not have input from the same place.

An Occurrence Net can be obtained from an ordinary place/transition net by an unfolding process as described in the following. At first place some initial tokens on the Petri Net, so to identify the initial marking of the net. Then for each of these tokens make a copy of the place in which they reside in the occurrence net, and then repeat the following process while there is a transition which can fire given the places introduced in the Occurrence Net.

- 1) Choose a transition from the Petri Net and call it t .
- 2) For each place in the pre-set of t , find a copy in the Occurrence Net and mark it with a token, if you can not find a copy, go back to step 1 (note - for a given t , do not choose the same subset of places in the Occurrence Net twice).
- 3) If any of the **places marked in the Occurrence Net are not concurrent**, go to step 1. Two places x and y are said to be concurrent if: (i) There is not path of arrows from x to y or vice versa; (ii) There is no third place z from which you can reach both x and y existing z by different arrow (this is called conflict).
- 4) If there is any transition t'' in the Occurrence Net such that **the local configuration is smaller but it has the same final state**, then mark t as a cut-off and go to step 1.
- 5) Make a copy of t in the Occurrence Net, call it t' , draw an arrow from every place you marked in the Occurrence Net to t' , erase the tokens.
- 6) For each place in the post-set of t , make a copy in the Occurrence Net, label it, and draw an arrow from t' to it.

As soon as the unfolding has been created we can answer the question of deadlock using branch and bound techniques that are suitable also for very large unfolding. The key observation is that there is no terminal marking exactly when every configuration of the unfolding is a subset of some configuration containing a cut-off point. There is a terminal marking if and only if there is a configuration, which is in conflict with every cut-off point. For example, if a configuration C' is in conflict with a cut-off point t' , there must be a transition $t'_1 \in C'$ which is in conflict with t' . Such transition t'_1 will be called a spoiler of t' . In other words there exists a terminal marking if and only if there exists a configuration containing a spoiler for every cut-off.

In order to show how the approach works, we consider the Petri Net of the basic example showed in Fig. 5.7 and we apply at first the algorithm and then branch and bound techniques for deadlock detection. In Fig. 5.11 we graphically show some steps and in Fig. 5.12 we propose the Occurrence Net resulting from the unfolding. In Table 5.1 we give the complete picture; for each step we propose the name of transition under selection, its local configuration together with the cardinality, we also show the labels in term of marking and we indicate if the selected transition is a cut-off or not. Table 5.2

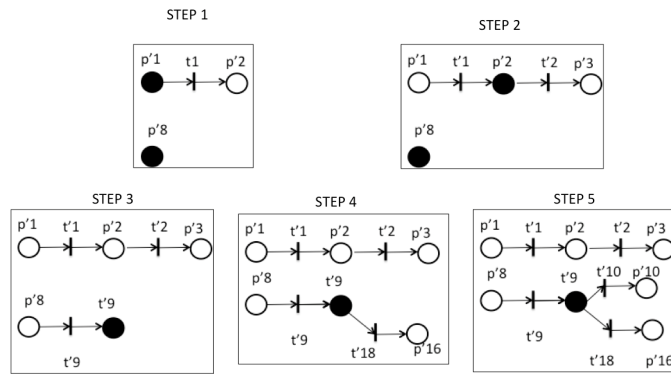


Figure 5.11: Unfolding a net: some steps.

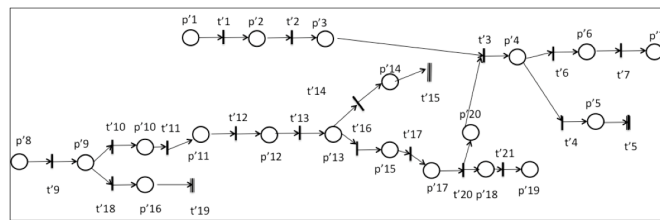


Figure 5.12: Unfolding a net: resulting OccurrenceNet.

reports the transitions that are in conflict with the cut-off, we observe that transition t_6 , t_7 and t_{21} are in conflict with each cut-off points t_5 , t_{15} and t_{19} . Considering that t_7 and t_{21} are final states for the BP we can say that t_6 is a deadlock, as it is also reported in the Table 5.1.

Step	Selected Transition	Local Configuration	Cardinality of Local Configuration	Marking	Cut-off	Deadlock
1	t_1	t'_1	1	p'_1	-	-
2	t_2	t'_1, t'_2	2	p'_3	-	-
3	t_9	t'_9	1	p'_9	-	-
4	t_{18}	t'_9, t'_{18}	2	p'_{16}	-	-
5	t_{10}	t'_9, t'_{10}	2	p'_{10}	-	-
6	t_{11}	t'_9, t'_{10}, t'_{11}	3	p'_{11}	-	-
7	t_{12}	$t'_9, t'_{10}, t'_{11}, t'_{12}$	4	p'_{12}	-	-
8	t_{13}	$t'_9, t'_{10}, t'_{11}, t'_{12}, t'_{13}$	5	p'_{13}	-	-
9	t_{14}	$t'_9, t'_{10}, t'_{11}, t'_{12}, t'_{13}, t'_{14}$	6	p'_{14}	-	-
10	t_{15}	$t'_9, t'_{10}, t'_{11}, t'_{12}, t'_{13}, t'_{14}, t'_{15}$	7	p'_{11}	yes	-
11	t_{16}	$t'_9, t'_{10}, t'_{11}, t'_{12}, t'_{13}, t'_{14}, t'_{15}, t'_{16}$	6	p'_{15}	-	-
12	t_{17}	$t'_9, t'_{10}, t'_{11}, t'_{12}, t'_{13}, t'_{14}, t'_{15}, t'_{16}, t'_{17}$	7	p'_{17}	yes	-
13	t_{19}	t'_9, t'_{18}, t'_{19}	3	p'_{17}	-	-
14	t_{20}	$t'_9, t'_{10}, t'_{11}, t'_{12}, t'_{13}, t'_{14}, t'_{15}, t'_{16}, t'_{17}, t'_{20}$	8	p'_{18}, p'_{20}	-	-
15	t_{21}	$t'_9, t'_{10}, t'_{11}, t'_{12}, t'_{13}, t'_{14}, t'_{15}, t'_{16}, t'_{17}, t'_{20}, t'_{21}$	9	p'_{19}	-	-
16	t_3	$t'_9, t'_{10}, t'_{11}, t'_{12}, t'_{13}, t'_{14}, t'_{15}, t'_{16}, t'_{17}, t'_{20}, t'_3$	11	p'_4	-	-
17	t_6	$t'_9, t'_{10}, t'_{11}, t'_{12}, t'_{13}, t'_{14}, t'_{15}, t'_{16}, t'_{17}, t'_{20}, t'_6$	12	p'_6	-	yes
18	t_7	$t'_9, t'_{10}, t'_{11}, t'_{12}, t'_{13}, t'_{14}, t'_{15}, t'_{16}, t'_{17}, t'_{20}, t'_7$	13	p'_7	-	-
19	t_4	$t'_9, t'_{10}, t'_{11}, t'_{12}, t'_{13}, t'_{14}, t'_{15}, t'_{16}, t'_{17}, t'_{20}, t'_4$	12	p'_5	-	-
20	t_5	$t'_9, t'_{10}, t'_{11}, t'_{12}, t'_{13}, t'_{14}, t'_{15}, t'_{16}, t'_{17}, t'_{20}, t'_5$	13	p'_3	yes	-

Table 5.1: Unfolding of the basic example.

Cut-off	Conflicts
t_5	$t_3, t_4, t_5, t_6, t_7, t_{16}, t_{17}, t_{20}, t_{21}$
t_{15}	$t_6, t_7, t_{14}, t_{15}, t_{18}, t_{19}, t_{21}$
t_{18}	$t_3, t_4, t_5, t_6, t_7, t_{10}, t_{11}, t_{12}, t_{13}, t_{14}, t_{15}, t_{16}, t_{17}, t_{20}, t_{21}$

Table 5.2: Transitions in conflict with the cut-off.

5.3. Further Developments

As we described in the previous section, the properties we verify are the ones that concern control flow correctness. In addition, we are considering to continue the research activity extending our framework with the possibility to verify also Data flow correctness and Compliance properties that are following described.

5.3.1. Data Flow Correctness

Data flow concerns the representation of Data Objects inside the BP model and their changes of status based on the execution of activities. Data flow correctness consists in preserving the following properties [67].

- **No missing data.** The data flow schema of a process model might cause missing data at run-time if a data object exists which can be read during run-time without having been written by any preceding activity or provided by the outside environment (i.e., by a start message). Fig. 5.13 is an example.

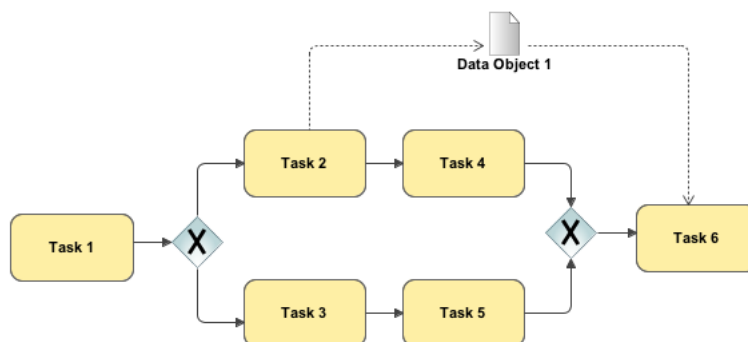


Figure 5.13: No missing data example.

- **Unnecessary data.** A data object written by an activity of process model is called unnecessary if it is not read by any subsequent activity or transition condition or passed to the outside environment via an end message. Fig. 5.14 is an example.

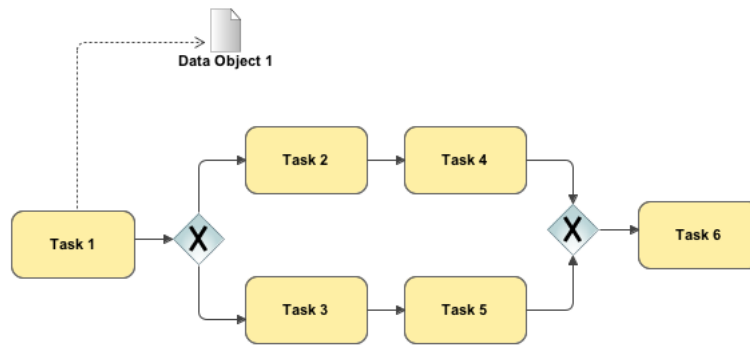


Figure 5.14: Unnecessary data example.

- **No lost updates.** The data flow schema of a process model might cause lost data at run-time if a data object, which is written by an activity, is updated by a subsequent activity, but without reading the data object in between. Fig. 5.15 is an example.

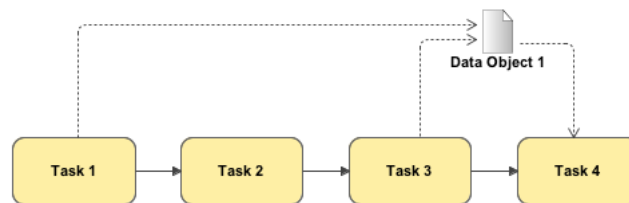


Figure 5.15: No lost updates example.

5.3.2. Compliance

Besides correctness checking, compliance is another relevant property. Generally, BP compliance concerns the entire process life cycle [67].

In the Learn PAd scenario we refer to a priori compliance checking; it verifies whether or not prespecified process models obey compliance rules. We also concentrate on a user-friendly approach considering all the different BP related perspectives included in the Learn PAd meta-model. The compliance properties we following define involve the control flow, the data flow, resource and time perspectives. It actually consists of compliance rules which may: impose constraints on the control flow schema of process models, constrain the data to be managed, require certain types of activities to be present in a process model, or enforce access control policies.

Extended Compliance Rule Graph

Several techniques are available to implement compliance. In Learn PAd we refer to Compliance Rule Graphs (CRGs) that are graphical notations that allow modelling compliance rules on a higher level of abstraction based on graphs [40]. More precisely CRG is an acyclic graph consisting of an *antecedence pattern* complemented by a *consequence pattern*. Both, the antecedence and the consequence pattern consist of occurrence and absence nodes. *Occurrence nodes* represent the occurrence of events of associated type and properties and edges constraining their ordering, while *absence nodes* represent the absence of certain events. The nodes are connected by directed edges that may also connect antecedence nodes with consequence nodes. Fig. 5.16 shows this basic elements of a CRG.

	ANTECEDENCE	CONSEQUENCE
OCCURENCE	A	B
ABSENCE	C	D

Figure 5.16: Nodes of CRG.

While CRG covers only the control flow perspective, the last works in the field of compliance checking use the extension of the CRG language, the extended Compliance Rule Graphs (eCRG), to support multi perspectives and specify requirements for cross-organizational scenarios [42]. This better fit with the Learn PAd meta-model. The eCRG language is based on CRG language. To cover various perspectives, it adds *attachments* to the nodes and connectors (i.e edges). They are constraints to the nodes or edges they are attached to. Depending on the included perspective, the eCRG offers different elements. In Appendix D it will be explained in detail.

Extended Compliance Rule Graph into Practice

In order to show how the approach works into practice we provide the application of the eCRG language to one BP included in the Learn PAs demonstrators, the “Sportello Unico Attività Produttive”, and in particular the “SCIA commerciale” case. We derived the compliance rules from the description of the case and then we represented them with the eCRG.

R1 (Fig. 5.17): Whenever task *Apply for open Business Activity* is performed by the entrepreneur, the data object *Istanza Scia* is sent and task *Wait confirmation* has to occur. Then task *Start Business Activity* is performed by the same entrepreneur.

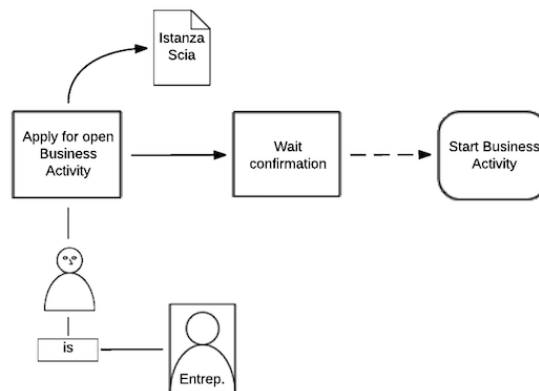


Figure 5.17: R1 eCRG.

R2 (Fig. 5.18): The entrepreneur can use an agency to perform his work through the data object *Delega*. Whenever an Agency receives *Delega* from an entrepreneur, it can perform task *Apply for open Business Activity*.

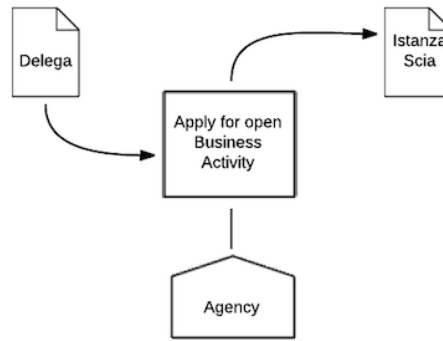


Figure 5.18: R2 eCRG.

R3 (Fig. 5.19): Whenever task *Apply for open Business Activity* is performed by the entrepreneur through the use of an Agency, the latter has to receive a confirmation, then it can inform the entrepreneur that he can start his activity.

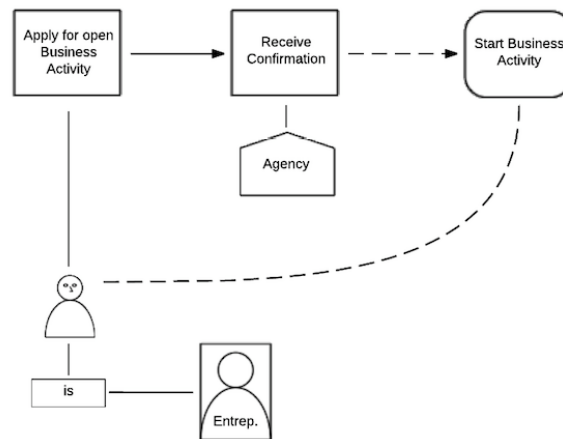


Figure 5.19: R3 eCRG.

R4 (Fig. 5.20): Whenever *Istanza Scia* is sent to SUAP Office, it is possible to require an integration to the entrepreneur within 30 days.

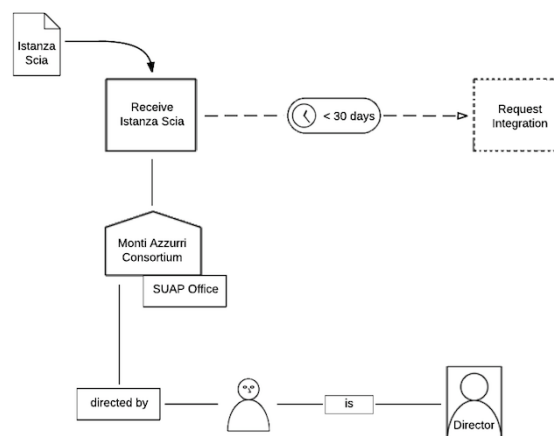


Figure 5.20: R4 eCRG.

R5 (Fig. 5.21): Whenever task *Request Integration* occurs and it is performed by SUAP Office, then task *Send Integration* has to occur and be performed by the entrepreneur.

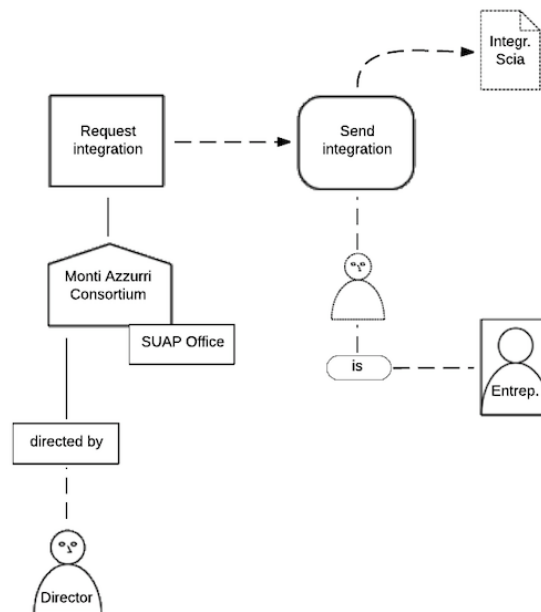


Figure 5.21: R5 eCRG.

R6 (Fig. 5.22): Task *Close Business Activity* must be performed by the entrepreneur. Prior to this, task *Send communication of close Business Activity* has to occur and be performed by SUAP Office, when it receives the data object *Verifica* from third party PAs and Municipality of Sarnano. The communication has to be received within 60 days from the dispatch of *Istanza Scia*. Task *Send communication of close Business Activity* is performed only when the check is negative.

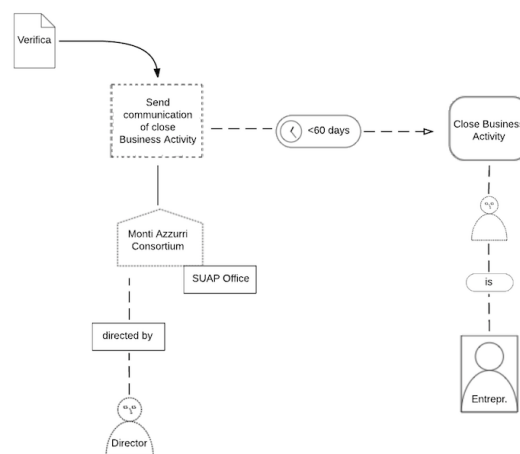


Figure 5.22: R6 eCRG.

R7 (Fig. 5.23): Whenever task *Receive request* occurs, with the arrive of the data object *Istanza Scia*, before task *Check Istanza Scia* performed by SUAP Office, the same office has to perform task *Send Confirmation* when all the documentation is right.

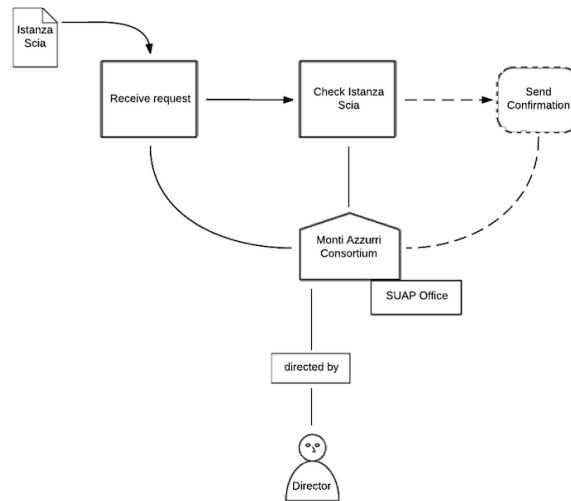


Figure 5.23: R7 eCRG.

R8 (Fig. 5.24): Whenever task *Check Istanza Scia* occurs and it is performed by SUAP Office, then task *Check Request's Acceptability* must be performed by the same office.

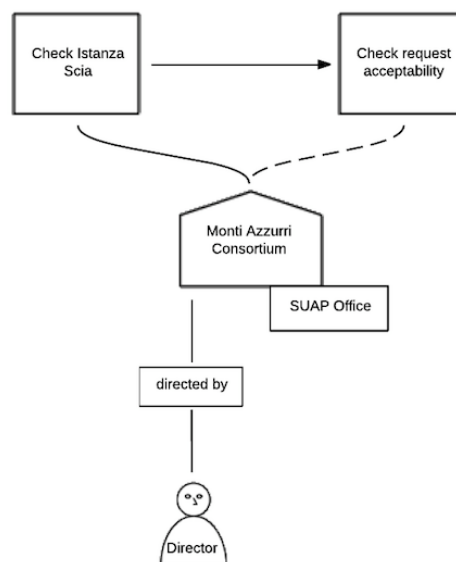


Figure 5.24: R8 eCRG.

R9 (Fig. 5.25): Whenever task *Check Request's Acceptability* is performed by the SUAP Office, then two option are possible.

- not admissible: task *Request Integration* is performed by SUAP Office.
- admissible: task *Forward Request* is performed by SUAP Office and *Istanza Scia* is forwarded.

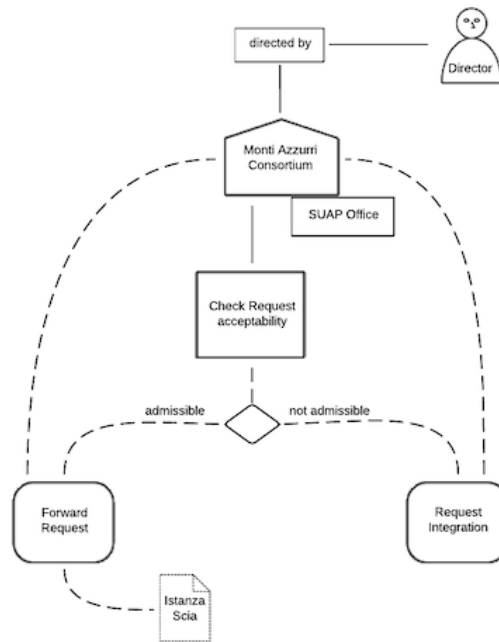


Figure 5.25: R9 eCRG.

R10 (Fig. 5.26): Whenever *Istanza Scia* is received by Third Parties, task *Analyze Istanza Scia* is performed by the same and the result *Verifica* is forwarded to SUAP Office.

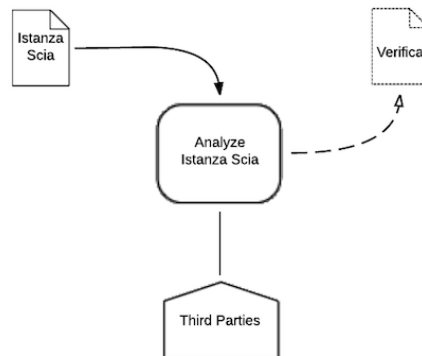


Figure 5.26: R10 eCRG.

R11 (Fig. 5.27): Whenever *Integrazione Scia* is received by Third Parties, the same offices can perform task *Send communication of reception*.

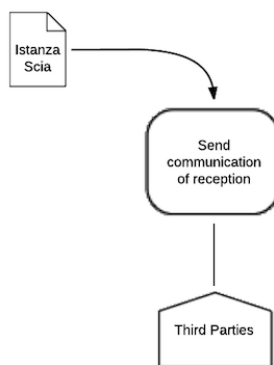


Figure 5.27: R11 eCRG.

R12 (Fig. 5.28): Whenever SUAP Office receives *Verifica*, then the same office can ask for other checks to the Municipality and perform task *Forward Istanza Scia to Municipality*.

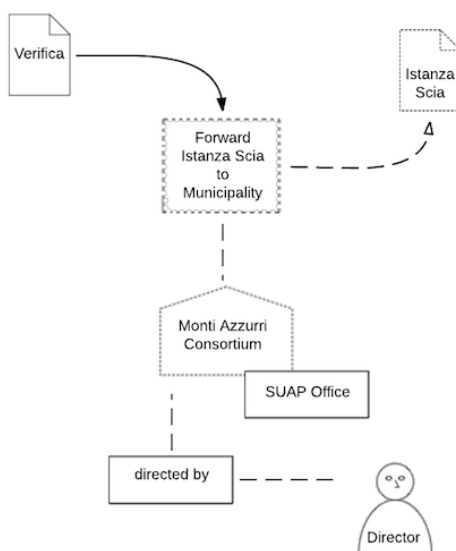


Figure 5.28: R12 eCRG.

R13 (Fig. 5.29): Whenever *Istanza Scia* arrived to Municipality, the Economic Office checks *Requisiti professional e morali*; and Urban Office checks *Disponibilità immobiliare*.

R14 (Fig. 5.30): Whenever *Integrazione Scia* is received by Municipality (Economic Office and Urban Office), the same offices can perform task *Send communication of reception*.

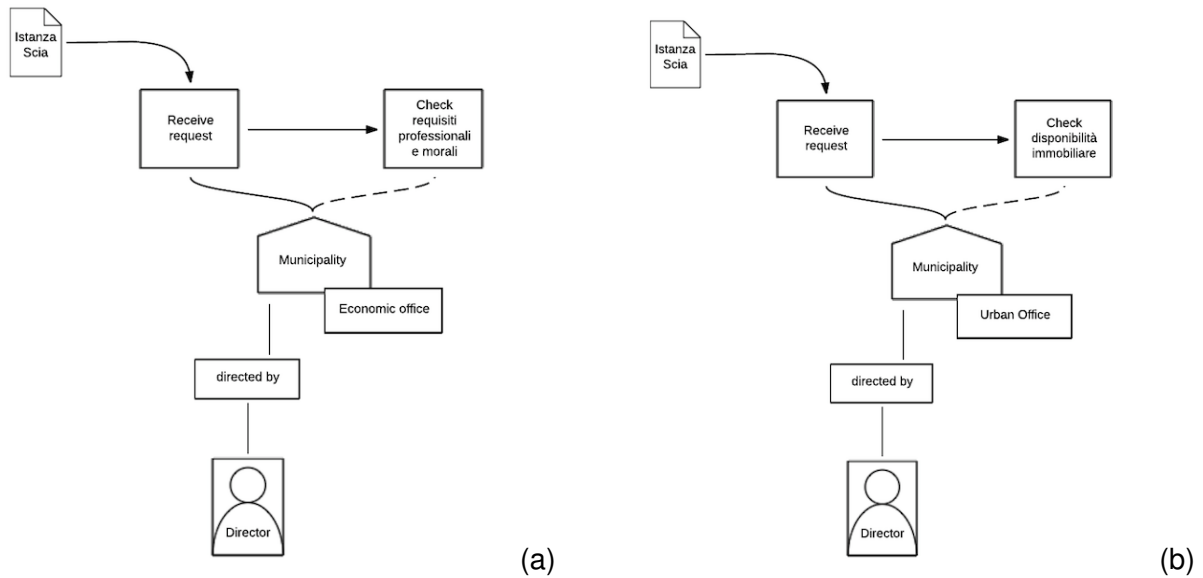


Figure 5.29: R13 eCRG: (a) Economic Office case and (b) Urban Office case .

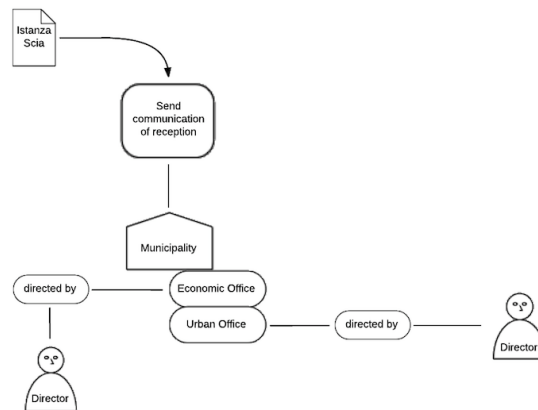


Figure 5.30: R14 eCRG.

Concluding, the rules we presented impact on control flow, data, resources and time perspectives. All of them are included in the Learn PAd meta-model. We just focus on the SUAP demonstrator considering “SCIA commerciale”; we plan to extend such compliance analysis also to the other BPs in the SUAP case.

6 Architectural View on Verification Component

In this chapter we provide a detailed description of the Verification component architecture. This component is engaged to manage all the aspects of the Learn PAd model formal analysis. In particular in this chapter we will describe all its sub-components, all the executions flows supported and the interfaces provided.

6.1. Verification Component Architecture Overview

In Fig. 6.1 a general overview of the Verification component architecture is provided. As can be seen by this overview, currently the architecture is mainly composed by 4 sub-components: the BPMN to Petri Net generator, the optimization engine, the model checker component and the metrics calculator.

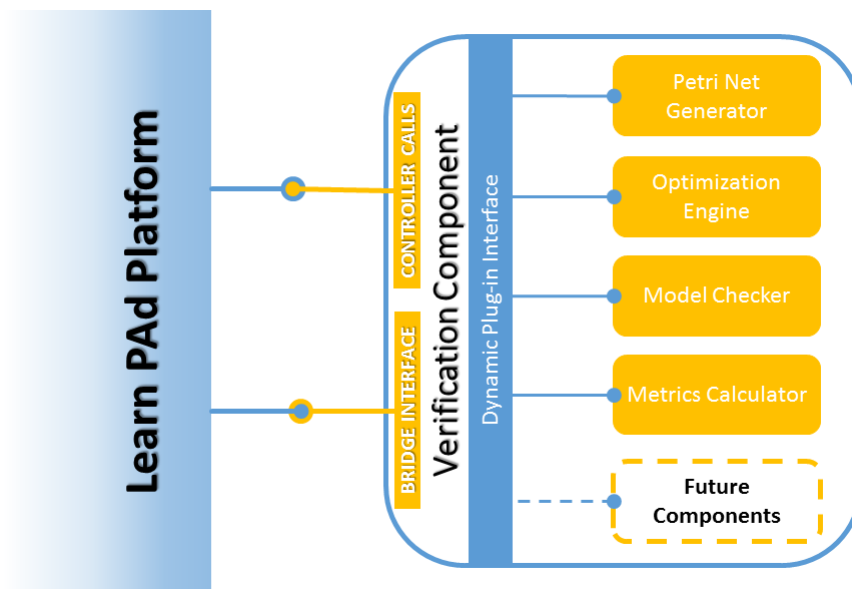


Figure 6.1: Verification Component Architecture

With this initial structure of the component we are able to formally verify structural problems, present in the flow of the modeled Business Process and to evaluate its understandability. In order to verify other aspects described in the Learn PAd model, this architecture can be extended adding other sub-components used to generate the relative Petri Net for each specific aspect. The optimization engine and the model checker component will be reused. Thanks to the pluggable interface implemented, each component can be substituted with another providing the same functionality and new modules can be added in order to perform other kind of verification checks. As any Learn PAd module, the Verification component will expose a set of functionalities to the platform through the bridge interface while will call functions inside its controller in order to contact the platform. Following we will discuss in detail each component.

6.1.1. Business Process to Petri Net Generator

Thanks to this sub-component, it is possible to create a Petri Net that strictly represent the Business Process flow of the Learn PAd model. This module integrates the mapping rules previously described and is the point of access for the Verification component as it will be described in the following sections. For each BPMN element this module will apply the right mapping rule creating the corresponding Petri Net elements. The generated Petri Net elements are then connected as described by the Sequence and Message Flow of the BPMN model. The component uses a specific data structure in order to represent, memorize and interact with a Petri Net. This structure is shared between all the components in order to simplify the operations of generating, optimizing Petri Nets and exporting them in a format accepted by the Model Checker.

Once a Petri Nets as been generated from the BPMN model, the optimization engine is called.

6.1.2. Optimization Engine

This module manage all the mechanisms that can be applied to a Petri Net in order to minimize it and optimize the work of the Model Checker component. All the algorithms applied in this phase have to preserve the original semantic of the Petri Net. In particular this module implements two optimization mechanisms:

- *Unfolding*: it will generate an unfolded net starting from the original one. The unfolding process has been described in the previous section 5.
- *Reduction*: it will apply some simple reduction rules in order to simplify the Petri Net preserving its original semantic. In this way it is possible to alleviate state space explosion in the model checking phase. This mechanism will be a next feature so a detailed description of the reduction rules will be provided in a future version of the document.

6.1.3. Model Checker Component

The model checker component is the engine that performs the formal verification of a specific property on the model. It relies on an external tool named LOLA (LOW Level petri net Analyzer) and provides mechanisms in order to interact with the tool in terms of:

- providing the Petri Net model in the specific syntax accepted by the tool.
- providing the property in the specific syntax accepted by the tool.
- processing the tool results for internal analysis.

LOLA is released as open source under GNU License. It is well consolidated and implement state of the art techniques in order to perform property verification in a formal and efficient way, as demonstrated by its score obtained on several Model Checking contests ¹.

The model checker component in particular take care of the following phases:

- 1) *Property Generation*: in this phase we generate a property strictly specific for the model to verify.
- 2) *Petri Net Model Export*: in this phase the internal Petri Net model is exported in the specific format accepted by the model checker tool. LOLA in particular uses its own format named EBNF in order to represent a Petri Net.
- 3) *Property Export*: in this phase we export the previously generated property in the syntactical format accepted by the model checker. The properties accepted by LOLA are expressed as CTL* formulas extended with some special words.

¹<http://mcc.lip6.fr/>

- 4) *Verification*: the module now calls the model checker tool and waits for a response. This part of the process can be really time consuming, so the component has to manage the process termination after a specific timeout in order to not saturate the machine resources. It also has to perform a multi-thread execution so it can be able to get, each time, the status of any process.
- 5) *Result Analysis*: in this phase the output of the model checker tool is captured and processed in order to return true when the property is verified, or false otherwise. This module will also analyze the counter example trace when the property is verified and reports it in a structured way.

6.1.4. Metrics Calculator

This module implements all the rules defined in the previous section 4 and is used to calculate understandability metrics for a given Business Process. This sub-component is the only one disjoint from the previously described because don't need translation in Petri Net and model checking but will work directly with the BPMN model analyzing its elements.

6.2. Interaction Flows

In this section we first provide a description of the interaction flows between the Verification component and the Learn PAd platform, then we provide a description of the interaction between the modules inside the Verification component.

6.2.1. External Flows

The Verification component is able to manage two kind of interactions with the Learn PAd Platform described in the Fig. 6.2 and 6.3.

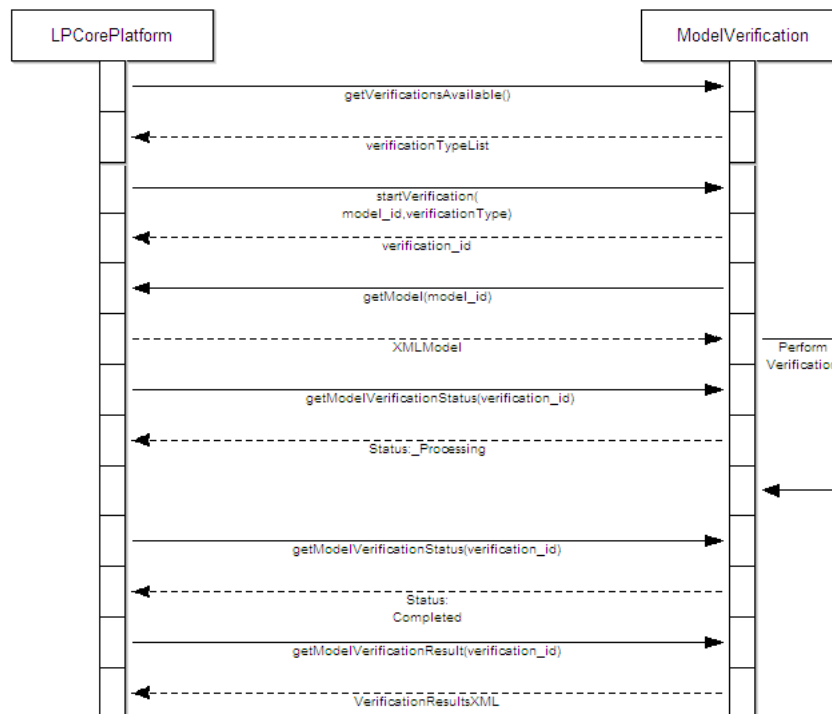


Figure 6.2: Sequence Diagram: Platform Verifier Scenario 1.

Fig. 6.2 describes a scenario where the Verification component has to be called in order to obtain the status of a specific verification on a model. The type of verification to be performed can be chosen

between all the available verification types provided by the component. The process is started by the platform that asks for the verification of a model. The model has to be present inside the platform. This action of the platform can be automatically invoked on every model imported, or can be manually invoked from the modeling environment that asks for a Verification. The request for verification returns a unique Id that has to be used in order to check the status of the verification. After this request the Verification component obtains the model from the platform and starts the verification. In any moment after the verification request, the platform can ask for the verification status and obtain a response that can be “In progress” or “Completed”. When the platform obtains the status “Completed” it can ask for the result that will be provided in XML format. If this last method is called when the process is still in progress, an error is returned.

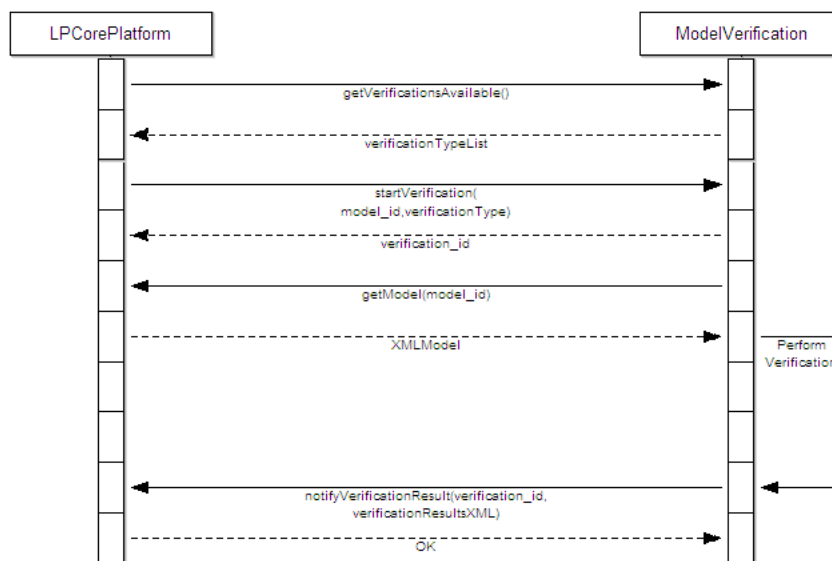


Figure 6.3: Sequence Diagram: Platform Verifier Scenario 2.

Fig. 6.3 describes instead a fully pushed scenario where the Verification component notifies the platform on verification completed. This flow starts in the same way of the previous one (described in Fig. 6.2), but in this case when the verification terminates the platform is instantly notified.

6.2.2. Internal Flows

In the section 6.1 we have described several components that work together in order to perform some kind of verification. In particular we have seen a division of two groups of components: one used to verify properties over a Petri Net model and one that work directly with the BPMN model. In this section we will describe the sequence of actions that will take part in each group.

- Fig. 6.4 provides a detailed view of the interaction between the Learn PAd Platform and the Verification Component with the scenario of Fig. 6.2 with in addition the interactions between all the sub-modules of the Verification Component involved in the verification of properties over a Petri Net model.

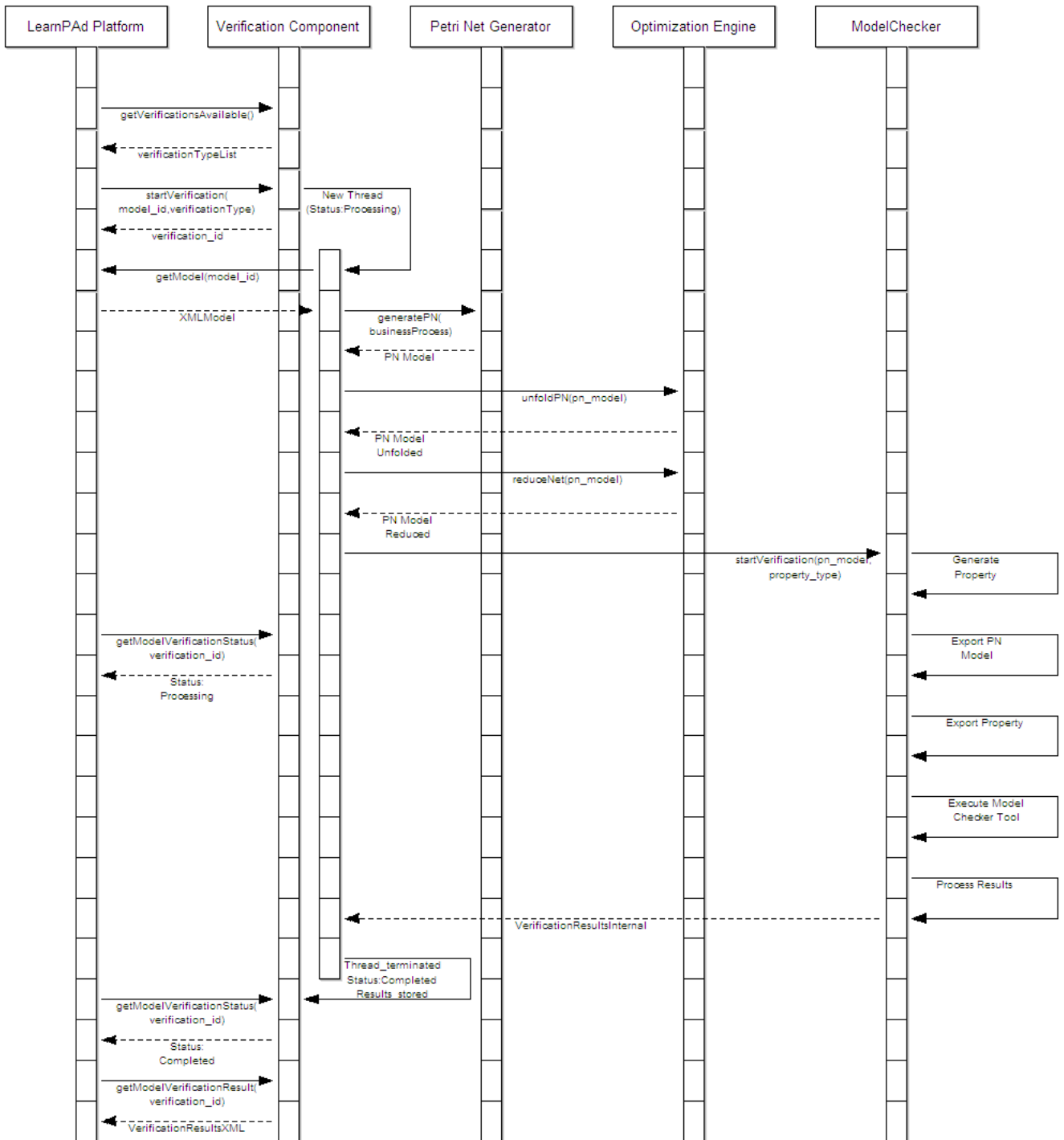


Figure 6.4: Sequence Diagram: Platform Verifier Internal Formal Scenario 1.

The process in this scenario is started by the platform that asks for the verification of a model. The type of verification to be performed can be chosen between all the available verification types provided by the component. In this case the type must be one related to property verification over Petri Net models like deadlock checks. The request for verification returns a unique Id that has to be used in order to check the status of the verification. After this request the Verification component generates a new verification thread referenced by the returned unique Id. This thread obtains the model from the platform and starts the verification generating a PN model from the BP. After that the optimization mechanisms are applied generating the unfolded net and reducing

the net. When all the optimization phases are completed the model checker module generates the property to verify, it exports the property and the model in the format accepted by the model checker tool and it executes the external tool. When the tool terminates its execution, the module processes the result and returns a response with counter example to the verification component. Received the response, the Verification component terminates the thread and it stores the result for future requests changing its status to completed. In any moment after the verification request, the platform can ask for the verification status. In this case the Verification component will look for the specified Thread and obtain a response that can be “In progress” if the thread is present or “Completed” if the thread is already terminated. When the platform obtains the status “Completed” it can ask for the result that will be provided in XML format.

- Fig. 6.5 provides a detailed view of the interaction between the Learn PAD Platform and the Verification Component with the scenario of Fig. 6.2 with in addition the interactions between the sub-module of the Verification Component involved in the verification of process understandability.

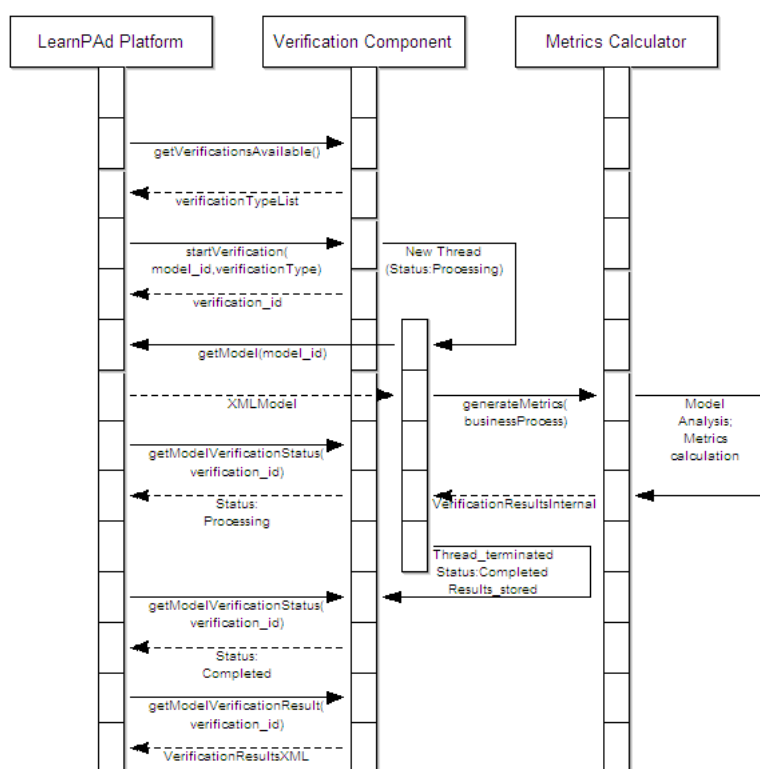


Figure 6.5: Sequence Diagram: Platform Verifier Internal Metrics Scenario 1.

The process in this scenario is started by the platform that asks for the verification of a model. The type of verification to be performed can be chosen between all the available verification types provided by the component. In this case the type must be one that specify the verification of understandability. The request for verification returns a unique Id that has to be used in order to check the status of the verification. After this request the Verification component generates a new verification thread referenced by the returned unique Id. This thread obtains the model from the platform ad starts the analysis calling the Metrics Calculator module. When this module finish its execution, it returns all the generated metrics to the verification component. Received the response, the Verification component terminates the thread and it stores the result for future requests changing its status to completed. In any moment after the verification request, the platform can ask for the verification status. In this case the Verification component will look for the specified Thread and obtain a response that can be “In progress” if the thread is present or “Completed” if

the thread is already terminated. When the platform obtains the status “Completed” it can ask for the result that will be provided in XML format.

7 Conclusions and Future Works

In this Deliverable, we presented first results of WP4 related to the analysis of BP models based on formal verification techniques to discover possible structural problems resulting from the model design-phase. The deliverable has the dual goals of providing quality characteristics suitable to support an effective learning in the Public Administration; and of supporting technical and research work-packages to better focus their activities with reference to the Learn PAd platform model verification component.

In Chapter 2 we present the Learn PAd quality assessment strategies for BP model with respect to the use of the platform. Different roles are introduced and discussed considering how each of them impacts on the platform.

In Chapter 3 we introduce some details related to the state of the art both in the area of BP model understandability and correctness. It helps the reader to better understand the contribution we provide.

In Chapter 4 we provide several guidelines collected in 4 categories (general, notation usage, labeling, and patterns). The guidelines relates to the goal of reaching a process model that can be understood by people. Understandability is considered a model external quality. Relevant is the impact on the learning since a BP model that is understandable can also be easily learned.

In Chapter 5 we discuss the need to formally check correctness for BP models. Indeed, in defining the notation, OMG did not provided a rigorous semantics for the various graphical elements; instead the meaning is given using natural language descriptions, permitting a wider adoption of the notation in different contexts. The use of formal tools to define the semantics of the various elements, and then of a BP model, is particularly interesting in order to enable automatic analysis activities that allow the designers to check if the BP satisfies desired properties or not. We consider a mapping from BPMN to Petri Net and we extensively describe properties that can be checked such as reachability, liveness, soundness, etc. We also consider unfolding techniques as a valuable approach to explore the state space of concurrent systems without considering all possible events interleaving. This makes the verification possible, avoiding the state explosion problem which is typical in verification processes.

In Chapter 6 we describe the results of the design phase related to the software quality assessment mechanisms included in the Learn PAd platform.

Based on the analysis of the state of the art and considering the contributions given in this Deliverable we plan to implement via automatic techniques (when possible) the BP model quality assessment. Both understandability and correctness will be considered. This is a pre-processing phase of the BP models that should enable the creation of wiki pages that are better understandable respect to those created without such quality assessment. This will be validated according to the Learn PAd validation strategies.

A Business Process Model Metrics

NAME	DESCRIPTION	SOURCE	YEAR
NT	Number of tasks.	[72]	2006
NCD	Number of complex decision.	[72]	2006
NDOin/NDOout	Number of data objects which are input/outputs of activities.	[72]	2006
NID	Number of inclusive decision.	[72]	2006
NEDDB	Number of exclusive data-based decision.	[72]	2006
NEDEB	Number of exclusive event-based decision.	[72]	2006
NL	Number of lanes.	[72]	2006
NMF	Number of message flows.	[72]	2006
NP	Number of pools.	[72]	2006
NPF	Number of parallel forking.	[72]	2006
NSFA	Number of sequence flows between activities.	[72]	2006
NSFE	Number of sequence flows from events.	[72]	2006
NSFG	Number of sequence flows from gateways.	[72]	2006
...	...	[72]	2006
CLA	Connectivity level between activities. Total Number of Activities / Number of Sequence Flows between these Activities. $CLA = TNA/NSFA$	[72]	2006
CLP	Connectivity level between participants. $CLP = NMF/NP$	[72]	2006
PDOPin/PDOPout	Proportion of data objects as incoming/outgoing products and total data objects. $PDOPIn = NDOIn/TNDO$; $PDOPOut = NDOOut/TNDO$	[72]	2006
TNT	Total number of tasks. $TNT = NT + NTL + NTMI + NTC$	[72]	2006
PDOTout	Proportion of data objects as outgoing product of activities of the model. $PDOTOut = NDOOut/TNT$	[72]	2006
PLT	Proportion of pools/lanes and activities $PLT = NL/TNT$	[72]	2006
TNCS	Total number of collapsed subprocesses. $TNCS = NCS + NCSL + NCSMI + NCSC + NCSA$	[72]	2006
TNA	Total number of activities. $TNA = TNT + TNCS$	[72]	2006
TNDO	Total number of data objects in the model. $TNDO = NDOIn + NDOOut$	[72]	2006
TNG	Total number of gateways. $TNG = NEDDB + NEDEB + NID + NCD + NPF$	[72]	2006
TNEE	Total number of end events. $TNEE = NENE + NEMsE + NEEE + NECaE + NECoE + NELE + NEMuE + NETE$	[72]	2006

Table A.1: Business Process Model Complexity Metrics. Part 1.

NAME	DESCRIPTION	SOURCE	YEAR
TNIE	Total number of intermediate events. $TNIE = NINE + NITE + NIMsE + NIEE + NICaE + NICoE + NIRE + NILE + NIMuE$	[72]	2006
TNSE	Total number of start events. $TNSE = NSNE + NSTE + NSMsE + NSRE + NSLE + NSMuE$	[72]	2006
TNE	Total number of events. $TNE = TNSE + TNIE + TNEE$	[72]	2006
CFC	Control-flow Complexity metric. It captures a weighted sum of all connectors that are used in a process model.	[10]	2005
NOA	Number of activities in a process.	[12]	2006
NOAC	Number of activities and control-flow elements in a process.	[12]	2006
NOAJS	Number of activities, joins, and splits in a process.	[12]	2006
HPC_D	Halsted-based Process Complexity (process difficulty).	[12]	2006
HPC_N	Halsted-based Process Complexity (process length).	[12]	2006
HPC_V	Halsted-based Process Complexity (process volume).	[12]	2006
NoI or Fan-in	Number of activity inputs. The fan-in of a procedure A is the number of local flows into procedure A plus the number of data structures from which procedure A retrieves information.	[12]	2006
NoO or Fan-out	Number of activity outputs. The fan-out of a procedure A is the number of local flows from procedure A plus the number of data structures which procedure A updates.	[12]	2006
Length	Activity length. The length is 1 if the activity is a black box; if it is a white box, the length can be calculated using traditional software engineering metrics that have been previously presented, namely the LOC (line of code) and MCC (McCabe's cyclomatic complexity).	[12]	2006
IC	Interface complexity of an activity metric. $IC = Length * (NoI * NoO)^2$, where the length of the activity can be calculated using traditional Software Engineering metrics such as LOC (1 if the activity source code is unknown) and NoI and NoO are the number of inputs and outputs.	[12]	2006
NOF	Number of control flow connections (number of arcs).	[12]	2006
TNSF	Total number of sequence flows.	[73]	2009
CC	Cross-connectivity metric. It is the ratio of the total number of arcs in a process model to the total number of its nodes.	[90]	2008
ICP	Imported Coupling of a Process metric. It counts, for each (sub-) process, the number of message/sequence flows sent by either the tasks of the (sub-) process or the (sub-) process itself.	[36]	2009
ECP	Exported Coupling of a Process metric. It counts, for each (sub-) process, the number of message/sequence flows received by either the tasks of the (sub-) process or the (sub-) process itself.	[36]	2009
W	Cognitive Weight. It measures the cognitive effort to understand a model; it can indicate that a model should be re-designed	[31]	2006
MaxND	Maximum Nesting Depth, where the nesting depth of an action is the number of decisions in the control flow that are necessary to perform this action.	[31]	2006

Table A.2: Business Process Model Complexity Metrics. Part 2.

NAME	DESCRIPTION	SOURCE	YEAR
MeanND	Mean Nesting Depth, where the nesting depth of an action is the number of decisions in the control flow that are necessary to perform this action.	[31]	2006
(Anti)Patterns for BPM	It counts the usage of anti-patterns. In a BP Model, it can help to detect poor modeling.	[31]	2006
CP	Coupling metric. The metric calculates the degree of coupling. Coupling is related to the number of interconnections among the tasks of a process model. The higher coupling value of the process, the more difficult it is to change the process and the higher probability that there will be errors in the process.	[69]	2004
Cohesion	Cohesion measures the coherence within the parts of the model.	[69]	2004
CNC	Coefficient of Network Complexity or Connectivity coefficient. It is the ratio of total number of arcs in a process model to its total number of nodes. It is calculated as: $CNC = NOF/NOAJS$.	[46]	2001
CI	Complexity Index (CI), or reduction complexity. It is defined as the minimal number of node reductions that reduces the graph to a single node.	[46]	2001
RT	Restrictiveness Estimator. It is an estimator for the number of feasible sequences in a graph. RT requires the reachability matrix r_{ij} , i.e. the transitive closure of the adjacency matrix, to be calculated. $RT = \frac{2\sum r_{ij} - 6(N-1)}{(N-2)(N-3)}$	[46]	2001
S_N	Number of nodes: number of activities and routing elements in a process model.	[52]	2008
$\Pi(G)$	Separability. It is the ratio of the number of cut-vertices divided by the total number of nodes in the process model.	[52]	2008
$\Xi(G)$	Sequentiality. It is the degree to which the model is constructed out of pure sequences of tasks. The sequentiality ratio is the number of arcs between none-connector nodes divided by the number of arcs.	[52]	2008
diam	Diameter. It is the length of the longest path from a start node to an end node.	[52]	2008
\wedge	Depth. It is the maximum nesting of structured blocks in a process model.	[52]	2008
GM or MM	Gateway Mismatch or Connector Mismatch. It is the sum of gateway pairs that do not match with each other, e.g. when an AND-split is followed up by an OR-join.	[52]	2008
GH or CH	Gateway Heterogeneity or Connector Heterogeneity. It defines the extent to which different types of connectors are used in a process model.	[52]	2008
Φ	Structuredness. It relates to how far a process model can be built by nesting blocks of matching join and split connectors. The degree of structuredness can be determined by applying reduction rules and comparing the size of the reduced model to the original size.	[52]	2008
CYC	Cyclicity. It captures the number of nodes in a cycle and relates it to the total number of nodes	[52]	2008

Table A.3: Business Process Model Complexity Metrics. Part 3.

NAME	DESCRIPTION	SOURCE	YEAR
TS or Concurrency	Token Splits or Concurrency. It captures the maximum number of paths in a process model that may be concurrently activate due to AND-splits and OR-splits; it sums up the output-degree of AND-joins and OR- joins minus one.	[52]	2008
$\Delta(G)$	Density. It is the ratio of the total number of arcs in a process model to the theoretically maximum number of arcs.	[52]	2008
ACD or AGD	Average Connector Degree or Average Gateway Degree. It is the average of the number of both incoming and outgoing arcs of the gateway nodes in the process model.	[53]	2007
MCD or MGD	Maximum Degree of a Connector or Maximum Gateway Degree. It is the maximum sum of incoming and outgoing arcs of these gateway nodes.	[53]	2007
ECaM	Extended Cardoso Metric. It is a Petri net version of metric that generalizes and improves the original CFC metric proposed by Cardoso. It focuses on the syntax of the model and ignores the complexity of the behavior.	[45]	2009
ECyM	Extended Cyclomatic Metric. It is directly adapted from McCabe Cyclomatic. It focuses on the resulting behavior and ignore the complexity of the model.	[45]	2009
SM	Structuredness Metric. It recognizes different kinds of structures in the process model and scores each structure by giving it some penalty value. The sum of these values is the Structuredness Metric (SM).	[45]	2009
DSM	Durfee Square Metric. It is based on h-index. It equals d if there are d types of elements which occur at least d times in the model (each), and the other types occur no more than d times (each)	[37]	2012
PSM	Perfect Square Metric. It is based on the g-index. Given a set of element types ranked in decreasing order of the number of their instances, the PSM is the (unique) largest number such that the top p types occur (together) at least p^2 times.	[37]	2012
Layout complexity	It evaluates the usability of different screen designs based on the Shannon formula.	[79]	1993
Layout appropriateness	It is the efficiency of a screen in terms of cost involved in completing a collection of tasks.	[13]	1996
Layout measure	It is a group of measures that quantify layout of models: number of edge crossing, number of non-rectilinear edges, overlapping area, etc.	[20]	2009

Table A.4: Business Process Model Complexity Metrics. Part 4.

B Business Process Verification Approaches

Caption bib	Title	Year of Publication	Process Modeling Language	Intermediate Process Formalization	Properties Modeling Language	Property Formalization	Verified Properties	Perspective	Scenario	Tool	Citations
[64]	The application of Petri nets to workflow management.	1998	WF-net	Petri-Net			Soundness	Control-Flow		Wolan	2827
[96]	Business process verification-finally a reality!	2009	YAWL	Petri-Net			Soundness, weak soundness, irreducible cancellation regions, and immutable OR-joins.	Control-Flow	visa application process for general skilled migration to Australia	YAWL workflow system	110
[88]	Verification of the SAP reference models using EPC reduction, state-space analysis, and invariants.	2007	EPC	Petri-Net			Soundness, Relaxed Soundness, and Invariant.	Control-Flow	Trade Department of a Dutch bank	Prom plug-in	75
[16]	Bridging the gap between business models and workflow specifications.	2004	EPC	Petri-Net			Relaxed Soundness and Robustness.	Control-Flow	Company incoming order.	Petrit and Wolan	162
[85]	Workflow verification: Finding control-flow errors using petri-net-based techniques.	2000	WF-net	Petri-Net			Soundness	Control-Flow		Wolan	468
[15]	Relaxed soundness of business processes.	2001	EPC	Petri-Net			Relaxed Soundness	Control-Flow	Handling of incoming goods	LOLA	136
[86]	Verification of workflow task structures: a Petri-net-based approach.	2000	Task Structure	Petri-Net			Soundness	Control-Flow		Wolan	306
[89]	Soundness and separability of workflow nets in the stepwise refinement approach.	2003	WF-nets				Soundness and separability.				125
[91]	Diagnosing workflow processes using Wolan.	2001	WF-nets				Soundness, Sateness, Deadness, and Liveness.		Travel Agency	Wolan	336
[97]	Transformation of BPMN to YAWL.	2008	BPMN	YAWL			Deadlock free, no dead task, proper completion, no or-join, soundness.			BPMN2YAWL open-source plug-in for Prom 5.0	22
[14]	Transforming BPMN Diagrams into YAWL Nets.	2008	BPMN	YAWL							33
[49]	A rigorous methodology for specification and verification of business processes.	2009	BPMN	Workflow Petri-Net	TLA+	LTL, CTL				TLC	6
[95]	Model checking of workflow schemas.	2000	WIMS	FSP			Safety and liveness.			TRACTA, LTS	80
[43]	From business process model to consistent implementation: a case for formal verification methods.	2002	Imperative	FSM		CTL	Termination, reachability, and fairness.			nuSMV	99
[69]	Model-checking behavioral specification of BPDL applications.	2006	BPDL	EFA to Promela		LTL			Loan Approval	SPIN	90
[93]	A Process-Algebraic Approach to Workflow Specification and Refinement.	2007	BPMN	CPS						FDR	67
[94]	A Process Semantics for BPMN.	2008	BPMN	CPS					airline ticket reservation	FDR	119
[95]	Verifying Business Process Compatibility (Short Paper).	2008	BPMN	CPS			compatibility			Haskell	22
[7]	A Method for Verifiable and Validatable Business Process Modeling.	2008	BPMN	Abstract State Machines method							39
[63]	Investigations on soundness regarding lazy activities.	2006	BPMN	pi-calculus			Lazy Soundness.			MWB and Advanced Bisimulation Checker.	77
[62]	Soundness Verification of Business Processes Specified in the Pi-Calculus.	2007	BPMN	pi-calculus			Soundness (easy, lazy, weak, relaxed, classical).			Advanced Bisimulation Checker	43
[48]	Enabling model checking for collaborative process analysis: from BPMN to Network of Timed Automata.	2015	BPMN 2.0	Network of Timed Automata		TCCTL	Interoperability			UPPAAL	
[34]	Model checking for managers.	1999	AMBER	Promela	Query Ptern	LTL	Sequences, combined occurrence or exclusion, and required precedence of activities.			SPIN	81

Table B.1: List of sources from the field of Business Process Verification. Part 1.

Caption bib	Title	Year of Publication	Process Modeling Language	Intermediate Process Formalization	Properties Modeling Language	Property Formalization	Verified Properties	Perspective	Scenario	Tool	Citations
[1]	Model checking for E-business control and assurance.	2005	UML Sequence Diagram	CSP		CSP			e-business protocol	FDR	24
[2]	Towards using reo for compliance-aware business process modeling.	2008	BPMN	Reo		Automata				Veredy/mCRL2	42
[4]	Efficient compliance checking using BPMN-Q and temporal logic.	2008	BPMN	Petri-Net	BPMN-Q	PLTL		Control flow	Bank account opening	LoLA/nuSMV	195
[5]	Specification, verification and explanation of violation for data aware compliance rules.	2009	BPMN		BPMN-W	PLTL		Control flow, data	Bank account opening		69
[17]	Automatic verification of data-centric business processes.	2009	Business artifacts			LTL-FO			customer purchase orders	Algorithms	168
[24]	Specification and verification of artifact behaviors in business process models.	2007	Business artifacts			ABS based on CTL-FO				Algorithms	107
[25]	Auditing business process compliance.	2007	Annotated BPMN	SPNets		Process effect rules				Algorithms	196
[26]	Designing compliant business processes with obligations and permissions.	2006	BPMN			temporal deontic logic, PENE-LOPE		Control flow, time		Algorithm for Prolog, PENELOPE in CLP(oid).	158
[28]	Compliance checking between business processes and business contracts.	2006	BPMN			FOL/NFCL		Control Flow	Service contract between ISP provider and a purchaser of a ISP services		236
[74]	Modeling control objectives for business process compliance.	2007	Simple language that can be mapped to BPMN			FOL		Control flow, data, resource, time	Purchase-to-pay		338
[27]	Detecting regulatory compliance for business process models through semantic annotations.	2009	BPMN			FOL		Control flow	account opening process in private banking		94
[50]	Verifying business processes using SPIN.	1998	Amber	Promela		LTL		Control flow, data		SPIN	52
[39]	On enabling data-aware compliance checking of business process models.	2010	BPMN			LTL		Control flow, data	Order-to-delivery scenario	plug-in of the Aristoflow BPM Suite	88
[40]	Ensuring business process compliance along the process life cycle.	2011	BPMN	YAWL	CRGs	LTL				Medical processes	11
[42]	Visual modeling of business process compliance rules with the support of multiple perspectives.	2013	BPMN		eCRGs	FOL		Control flow, time, data, resource	cross-organizational scenarios	Proof-of-concept implementation	19
[41]	An Operational Semantics for the Extended Compliance Rule Graph Language.	2014	BPMN	YAWL				Control flow, time, data, resource			1
[80]	Modeling the Resource Perspective of Business Process Compliance Rules with the Extended Compliance Rule Graph.	2014			eCRGs			Control flow, Time, Data, and Resource.	Healthcare		5
[47]	Design and verification of instantiable compliance rule graphs in process-aware information systems.	2010	BPMN		SeaFlows compliance rule formalism	FOL			Software development	SeaFlows compliance rule editor.	64
[56]	Abductive logic programming as an effective technology for the static verification of declarative business processes.	2010	ConDec graphical notation			LTL				SCIFF	16
[64]	Developer-friendly verification of process-based systems.	2010	EPC	Kripke Structure	EG-CTL	ECTL1			e-procurement system		16

Table B.2: List of sources from the field of Business Process Verification. Part 2.

C Background on Petri Net and Occurrence Net

Petri Net for Process Modelling

Petri Nets (PNs) are a graphical and mathematical modeling language for the description of concurrent and distributed systems [61]. A PN is a directed bipartite graph, in which the nodes represent both transitions and places. The directed arcs describe which places are pre- and/or post-conditions for which transitions (represented by arrows) occurs.

Definition A Petri Net is a triple $N = (P, T, F)$ where:

1. $P = \{p_1, p_2, \dots, p_m\}$ is the set of places;
2. $T = \{t_1, t_2, \dots, t_n\}$ is the set of transitions;
3. F is a subset of $(P \times T) \cup (T \times P)$ and it is referred as the flow relation. In case a couple (p, t) or (t, p) is in F we will indicate this fact with the infix notation pFt or tFp respectively.

The pre-set of transition t , denoted $\bullet t$, is the set of places p such that pFt . The post-set of transition t , denoted $t\bullet$, is the set of places p such that tFp . A marking M of a net (P, T, F) is a function $M : P \rightarrow \mathbb{N}$. We identify a marking M with the multi-set containing $M(p)$ copies of p for every $p \in P$. A tuple $\Sigma = (P, T, F, M_0)$ is a net system if (P, T, F) is a PN and M_0 is a marking of (P, T, F) called the initial marking of Σ . A marking M enables a transition t if $\forall p \in \bullet t : M(p) \geq 1$. If t is enabled at M , then it can occur and its occurrence leads to a new marking M' (denoted $M \xrightarrow{t} M'$), defined by $M' = M - \bullet t + t\bullet$. A sequence of transition $\sigma = t_1, t_2, t_3, \dots, t_n$ is an occurrence sequence if there exist marking $M_1, M_2, M_3, \dots, M_n$. M is a reachable marking if there exists an occurrence sequence σ such that $M_0 \xrightarrow{\sigma} M$.

We can classify a PN according to the capability of its places. A place in a Petri net is called k – bounded if it does not contain more than k tokens in all reachable markings, including the initial marking; it is said to be safe if it is 1 – bounded; it is bounded if it is k – bounded for some k . A (marked) Petri Net is called k – bounded, safe, or bounded when all of its places are k – bounded, safe, or bounded respectively. A net system is called deadlock-free if for every reachable marking at least one transition is enabled.

Occurrence Net for Unfolding

A particular type of Petri Net is referred as Occurrence Net. In this case the net will have no backward conflict and no cycle. In other words the occurrence net is a specialized form of net that must satisfy certain restrictions. At first, it must be well funded, meaning that the arrows cannot be followed backward infinitely from any point. Second, it must have no forward conflict, meaning that two arrows may not converge on the same place. Third, no event may be in conflict with itself, and fourth, no two events with the same label may have the same pre-set.

Definition A (P, T) Labelled Occurrence Net N' consist of a Petri Net (P', T', F') and a labeling function L' which maps P' onto a set P and T' onto a set T . The net must have the following properties:

1. **Well foundedness:** every subset of T' must contain a minimal element with respect to F'^* ;

2. **No forward conflict:** for all place $p \in P'$, $p \in t_1 \bullet$ and $p \in t_2 \bullet$ implies $t_1 = t_2$;
3. **No self-conflict:** for all $t_1, t_2, t_3 \in T'$ $t_1 F' * t_3$ and $t_2 F' * t_3$ and $\bullet t_1 \cup \bullet t_2 \neq \emptyset$;
4. **No redundancy:** for all $t_1, t_2 \in T'$, $L'(t_1) = L'(t_2)$ and $\bullet t_1 = \bullet t_2$ implies $t_1 = t_2$.

The unfolding of a Petri Net leads to a Labelled Occurrence Net. It allows to choose a partial rather than a total order on moves, and thereby avoid unnecessary bifurcation in the search. From the unfolding results that (i) pre-set and post-set of any transition in the unfolding match the pre-set and post-set of the corresponding transition in the original Petri Net and (ii) labels of places in the unfolding with no predecessors matches the initial marking of the original Petri Net.

In order to define the unfolding more details on the labeled occurrence net are needed. Following we consider configuration and in particular local configuration. We also introduce the definition of cut-off point.

Definition Let N' be a labeled occurrence net, a subset S of T' is a **configuration** of N' exactly when:

1. It is **backward closed:** if $t_1 \bullet t_2$, then $t_2 \in S$ implies $t_1 \in S$;
2. It is **conflict free:** for all distinct $t_1 \in S$, $t_2 \in S$, $\bullet t_1$ and $\bullet t_2$ are disjoint.

We can associate each configuration of the unfolding with a state of the original net by simply identifying those places whose tokens are produced but not consumed by the transition in the configuration. We have to underline that the order of the transitions in a configuration is given by the firing sequence. Thus a configuration has a well-defined final state represented by labels of marking on the original net. This final state is determined by the post-set of the configuration: those place on the frontier between events in the configuration and events not in the configuration. In other words, the final states of all the configurations are exactly the reachable marking of the original net.

The local configuration associated with any transition consists of that transition and all of its predecessors in the dependency order. A local configuration is a backward closure of any transition.

Definition Let N' be a labelled Occurrence Net, and let $t' \in T'$, the **local configuration** of t' , denoted as $\lceil t' \rceil$, is the least backward closed subset of T' , with respect to F , containing t' .

This is the set of transitions, which necessarily are contained in any configuration where it is possible to find the given transition. A local configuration has of course a final state. This help us to introduce the notion of cut-off point suitable to produce a truncated unfolding. This is possible once it has been established that the configuration of the infinite unfolding represents exactly the set of reachable markings of the original net, so a finite fragment of the unfolding might be constructed which is sufficient to represent all the reachable markings. A transition is identified as a cut-off point if there exists another transition whose local configuration is smaller, but it has the same final state.

Definition Let N' be an unfolding of a Petri Net N , a transition $t' \in T'$ is a **cut-off point** of N' exactly when there exists $t'' \in T'$ such that

- 1) $|\lceil t' \rceil| < |\lceil t'' \rceil|$;
- 2) $F(\lceil t' \rceil) = F(\lceil t'' \rceil)$.

Any configuration containing a cut-off point do not add new reachable marking to the unfolding, and then it can be excluded.

D extended Compliance Rule Graphs

In the following we introduce the notation introduced by extended Compliance Rule Graphs we use [42]. Different perspective are influenced by the notation.

The most relevant perspective is the control flow perspective. It offers elements for expressing both the occurrence or not of tasks as well as their ordering. Since eCRG is based on CRG, there are the same four different task elements (i.e. antecedence occurrence, antecedence absence, consequence occurrence, and consequence absence task), that allow expressing whether or not particular tasks must be executed. Fig. D.1 shows the elements of this perspective.

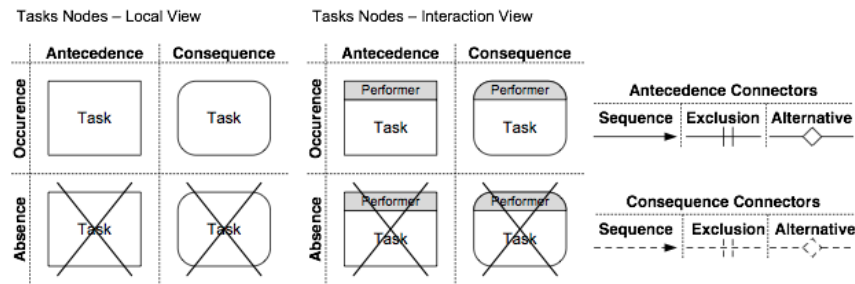


Figure D.1: Elements of the control flow perspective.

In addition to the sequence flow connector, other two types of connector are provided: the *exclusive connector* and the *alternative connector*. The first denotes mutual exclusion of tasks. The latter expresses that at least one of the connected tasks must occur. Exclusive as well as alternative connectors may only connect nodes that are both part of either the antecedence or consequence pattern. Note that the absence of sequence flow indicates parallel flow.

The interaction perspective covers constraints on exchanged messages in a cross-organizational scenario, so the interactions between the various partners. The message exchange is expressed in terms of particular nodes that reflect the events of sending and receiving a message. In addition, a message flow denotes the dependency between the events representing the sending and the receiving of a particular message, as shown in Fig. D.2.

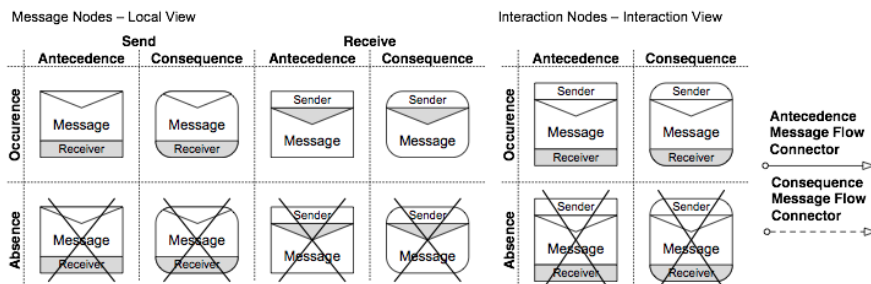


Figure D.2: Elements of the interaction perspective.

The elements for the time perspective can be divided into three types, as shown in Fig. D.3: *Points in Time*, *Time Conditions* and *Time Distance Connectors*.

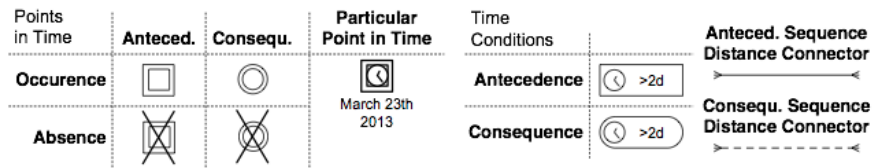


Figure D.3: Elements of the time perspective.

Points in Time nodes are used to express a particular date or point-in-time (e.g. 23th March 2013). *Time Conditions* concern the duration of a task. They may be attached to task nodes as well as sequence or message flow connectors to either constrain the duration of a task or the time distance between tasks, messages, and points in time. Finally *Time Distance Connectors* allow constraining the time distance without implying a particular sequence. They must be attached with a time condition.

The resource perspective requires concepts for expressing constraints on resources. It covers the different kinds of human resources as well as their inter-relations, and it allows constraining the assignment of resources to tasks. In particular, we consider resources like staff member, role, group, and organizational unit, and their relation to tasks. Furthermore, the selected approach supports resource conditions and relations among resources.

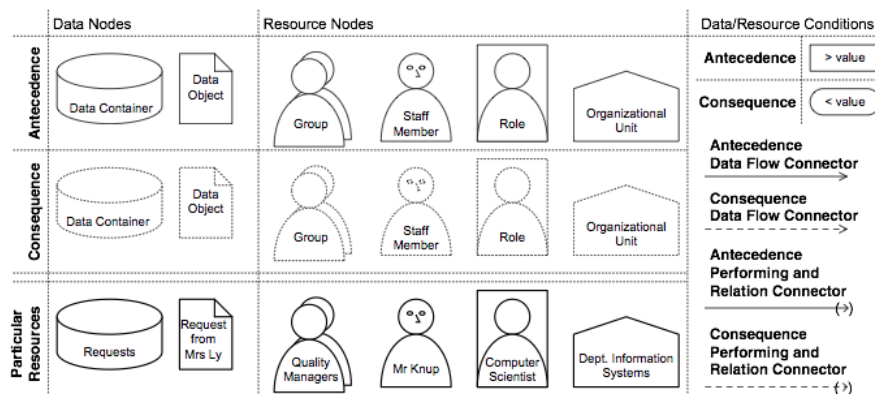


Figure D.4: Elements of resource and data perspective.

As shown in Fig. D.4, similar to task nodes, resource nodes may be part of the antecedence or consequence pattern. Alternatively, they may represent a particular resource instance (e.g. staff member Mr. Smith, or role supervisor). In turn, resource conditions may constrain resource nodes. Further, the performing relation indicates the performer of a task. Finally, resource relation connectors express relations between resources. Note that the resource perspective can be easily extended with other kinds of resources if required.

Fig. D.4 shows also elements of the data perspective. It comprises concept for express data-aware compliance rules. As for resource perspective, it is possible to find four types of elements: *Data Nodes*, *Data Conditions*, *Data Flow Connectors* and *Data Relation Connectors*. *Data Nodes* are of two types, *Data Container* and *Data Object*. The first refers to process data elements or global data stores. The latter instead, refers to particular data values and object instances. *Data Conditions* are used to constrain data container, data objects, and data flow. *Data Flow Connectors* define which process tasks read or write which data objects or data container. Finally, *Data Relation Connectors* may be used either to compare different data objects or to constrain the value of data containers at particular points in time.

Bibliography

- [1] Bonnie Brinton Anderson, James V. Hansen, Paul Benjamin Lowry, and Scott L. Summers. Model checking for E-business control and assurance. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 35(3):445–450, 2005.
- [2] Farhad Arbab, Natallia Kokash, and Sun Meng. Towards Using Reo for Compliance-Aware Business Process Modeling. In *Leveraging Applications of Formal Methods, Verification and Validation*, volume 17, pages 108–123. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.
- [3] Ahmed Awad, Gero Decker, and Niels Lohmann. Diagnosing and Repairing Data Anomalies in Process Models. In *Business Process Management Workshops*, number 43 in Lecture Notes in Business Information Processing, pages 5–16. Springer Berlin Heidelberg, 2010.
- [4] Ahmed Awad, Gero Decker, and Mathias Weske. Efficient compliance checking using bpmn-q and temporal logic. In *Business Process Management*, pages 326–341. Springer, 2008.
- [5] Ahmed Awad, Matthias Weidlich, and Mathias Weske. Specification, verification and explanation of violation for data aware compliance rules. In *Service-Oriented Computing*, pages 500–515. Springer, 2009.
- [6] Jörg Becker, Michael Rosemann, and Christoph von Uthmann. Guidelines of Business Process Modeling. In *Business Process Management*, volume 1806, pages 30–49. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.
- [7] Egon Börger and Bernhard Thalheim. A Method for Verifiable and Validatable Business Process Modeling. In *Advances in Software Engineering*, volume 5316, pages 59–115. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.
- [8] Elisabeth Bosshart, Mike Märki, Beat Rigert, Nicki Spöcker, and Christian Tanner. eCH-0158 BPMN-Modellierungskonventionen für die öffentliche Verwaltung. 2014.
- [9] Roberto Bruni, Andrea Corradini, Gianluigi Ferrari, Tito Flagella, Roberto Guanciale, and Giorgio Spagnolo. Applying Process Analysis to the Italian eGovernment Enterprise Architecture. In *Web Services and Formal Methods*, number 7176 in Lecture Notes in Computer Science, pages 111–127. Springer Berlin Heidelberg, 2012.
- [10] Jorge Cardoso. Control-flow complexity measurement of processes and weyuker’s properties. In *6th International Enformatika Conference*, volume 8, pages 213–218, 2005.
- [11] Jorge Cardoso. Process control-flow complexity metric: An empirical validation. In *Services Computing, 2006. SCC’06. IEEE International Conference on*, pages 167–173. IEEE, 2006.
- [12] Jorge Cardoso, Jan Mendling, Gustaf Neumann, and Hajo A. Reijers. A discourse on complexity of process models. In *Business process management workshops*, pages 117–128. Springer, 2006.
- [13] Tim Comber and John R. Maltby. Investigating layout complexity. 1996.

- [14] Gero Decker, Remco Dijkman, Marlon Dumas, and Luciano García-Bañuelos. Transforming BPMN diagrams into YAWL nets. In *Business Process Management*, pages 386–389. Springer, 2008.
- [15] Juliane Dehnert and Peter Rittgen. Relaxed Soundness of Business Processes. In *Advanced Information Systems Engineering*, volume 2068, pages 157–170. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.
- [16] Juliane Dehnert and Wil M. P. Van Der Aalst. Bridging the gap between business models and workflow specifications. *International Journal of Cooperative Information Systems*, 13(03):289–332, September 2004.
- [17] Alin Deutsch, Richard Hull, Fabio Patrizi, and Victor Vianu. Automatic verification of data-centric business processes. In *Proceedings of the 12th International Conference on Database Theory*, pages 252–267. ACM, 2009.
- [18] Remco M. Dijkman, Marlon Dumas, and Chun Ouyang. Semantics and analysis of business process models in BPMN. *Information and Software Technology*, 50(12):1281–1294, November 2008.
- [19] John Doe. *Modeling Guidelines Manual*. Signavio, 2014.
- [20] Holger Eichelberger and Klaus Schmid. Guidelines on the aesthetic quality of UML class diagrams. *Information and Software Technology*, 51(12):1686–1698, 2009.
- [21] Javier Esparza and Keijo Heljanko. *Unfoldings: a partial-order approach to model checking*. Springer Science & Business Media, 2008.
- [22] Damiano Falcioni, Andrea Polini, Alberto Polzonetti, and Barbara Re. Direct Verification of BPMN Processes through an Optimized Unfolding Technique. pages 179–188. IEEE, August 2012.
- [23] Michael Fellman and Andrea Zasada. State-of-the-Art of Business Process Compliance Approaches: A Survey. In *Proceedings of the 22nd European Conference on Information Systems (ECIS)*, 2014.
- [24] Cagdas Evren Gerede and Jianwen Su. *Specification and verification of artifact behaviors in business process models*. Springer, 2007.
- [25] Aditya Ghose and George Koliadis. *Auditing business process compliance*. Springer, 2007.
- [26] Stijn Goedertier and Jan Vanthienen. Designing compliant business processes with obligations and permissions. In *Business Process Management Workshops*, pages 5–14. Springer, 2006.
- [27] Guido Governatori, Jörg Hoffmann, Shazia Sadiq, and Ingo Weber. Detecting regulatory compliance for business process models through semantic annotations. In *Business Process Management Workshops*, pages 5–17. Springer, 2009.
- [28] Guido Governatori, Zoran Milosevic, and Shazia Sadiq. Compliance checking between business processes and business contracts. pages 221–232. IEEE, October 2006.
- [29] Guido Governatori, Francesco Olivieri, Simone Scannapieco, and Matteo Cristani. Designing for compliance: Norms and goals. In *Rule-Based Modeling and Computing on the Semantic Web*, pages 282–297. Springer, 2011.
- [30] Heerko Groefsema and Doina Bucur. A survey of formal business process verification: From soundness to variability. pages 198 – 203, 2013.
- [31] Volker Gruhn and Ralf Laue. Complexity metrics for business process models. In *9th international conference on business information systems (BIS 2006)*, volume 85, pages 1–12. Citeseer, 2006.

- [32] James A. Hanley and Barbara J. McNeil. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143(1):29–36, 1982.
- [33] Wenjia Huai, Xudong Liu, and Hailong Sun. Towards Trustworthy Composite Service Through Business Process Model Verification. In *2010 7th International Conference on Ubiquitous Intelligence Computing and 7th International Conference on Autonomic Trusted Computing (UIC/ATC)*, pages 422–427, October 2010.
- [34] Wil Janssen, Radu Mateescu, Sjouke Mauw, Peter Fennema, and Petra van der Stappen. Model Checking for Managers. In *Theoretical and Practical Aspects of SPIN Model Checking*, volume 1680, pages 92–107. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999.
- [35] Christos Karamanolis, Dimitra Giannakopoulou, Jeff Magee, and Stuart M. Wheeler. Model checking of workflow schemas. In *Enterprise Distributed Object Computing Conference, 2000. EDOC 2000. Proceedings. Fourth International*, pages 170–179. IEEE, 2000.
- [36] Wiem Khelif, Lobna Makni, Nahla Zaaboub, and Hanene Ben-Abdallah. Quality metrics for business process modeling. In *Proceedings of the 9th WSEAS international conference on Applied computer science*, pages 195–200. World Scientific and Engineering Academy and Society (WSEAS), 2009.
- [37] Krzysztof Kluza and Grzegorz Jacek Nalepa. Proposal of square metrics for measuring business process model complexity. In *Computer Science and Information Systems (FedCSIS), 2012 Federated Conference on*, pages 919–922. IEEE, 2012.
- [38] Krzysztof Kluza, Grzegorz Jacek Nalepa, and Janusz Lisiecki. Square Complexity Metrics for Business Process Models. In *Advances in Business ICT*, volume 257, pages 89–107. Springer International Publishing, Cham, 2014.
- [39] David Knuplesch, Linh Thao Ly, Stefanie Rinderle-Ma, Holger Pfeifer, and Peter Dadam. On enabling data-aware compliance checking of business process models. In *Conceptual Modeling—ER 2010*, pages 332–346. Springer, 2010.
- [40] David Knuplesch and Manfred Reichert. Ensuring business process compliance along the process life cycle. 2011.
- [41] David Knuplesch and Manfred Reichert. An Operational Semantics for the Extended Compliance Rule Graph Language. 2014.
- [42] David Knuplesch, Manfred Reichert, Linh Thao Ly, Akhil Kumar, and Stefanie Rinderle-Ma. Visual modeling of business process compliance rules with the support of multiple perspectives. In *Conceptual Modeling*, pages 106–120. Springer, 2013.
- [43] Jana Koehler, Giuliano Tirenni, and Santhosh Kumaran. From business process model to consistent implementation: A case for formal verification methods. In *Enterprise Distributed Object Computing Conference, 2002. EDOC’02. Proceedings. Sixth International*, pages 96–106. IEEE, 2002.
- [44] Ryszard Koniewski, Andrzej Zielinski, and Krzysztof Amborski. Use of Petri Nets and Business Processes Management Notation in Modelling and Simulation of Multimodal Logistics Chains. In *Proceedings 20th European Conference on Modeling and Simulation, Institute of Control and Industrial Electronics, Warsaw University of Technology*, 2006.
- [45] Kristian Bisgaard Lassen and Wil MP van der Aalst. Complexity metrics for Workflow nets. *Information and Software Technology*, 51(3):610–626, 2009.
- [46] Antti M. Latva-Koivisto. Finding a complexity measure for business process models. *Helsinki University of Technology, Systems Analysis Laboratory*, 2001.

- [47] Linh Thao Ly, Stefanie Rinderle-Ma, and Peter Dadam. Design and verification of instantiable compliance rule graphs in process-aware information systems. In *Advanced Information Systems Engineering*, pages 9–23. Springer, 2010.
- [48] Sihem Mallek, Nicolas Daclin, Vincent Chapurlat, and Bruno Vallespir. Enabling model checking for collaborative process analysis: from BPMN to ‘Network of Timed Automata’. *Enterprise Information Systems*, 9(3):279–299, April 2015.
- [49] Cristian Masalagiu, Wei-Ngan Chin, Ștefan Andrei, and Vasile Alaiba. A rigorous methodology for specification and verification of business processes. *Formal Aspects of Computing*, 21(5):495–510, October 2009.
- [50] Sjouke Mauw, Radu Mateescu, and Wil Janssen. Verifying business processes using SPIN. In *Proceedings of the International SPIN Workshop*, pages 21–36, 1998.
- [51] Kenneth L. McMillan and David K. Probst. A technique of state space search based on unfolding. *Formal Methods in System Design*, 6(1):45–65, 1995.
- [52] Jan Mendling. Metrics for Business Process Models. In *Metrics for Process Models*, volume 6, pages 103–133. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.
- [53] Jan Mendling, Hajo A. Reijers, and Jorge Cardoso. What makes process models understandable? In *Business Process Management*, pages 48–63. Springer, 2007.
- [54] Jan Mendling, Hajo A. Reijers, and Wil MP van der Aalst. Seven process modeling guidelines (7pmg). *Information and Software Technology*, 52(2):127–136, 2010.
- [55] Jan Mendling, Laura Sanchez-Gonzalez, Felix Garcia, and Marcello La Rosa. Thresholds for error probability measures of business process models. *Journal of Systems and Software*, 85(5):1188–1197, 2012.
- [56] Marco Montali, Paolo Torroni, Federico Chesani, Paola Mello, Marco Alberti, and Evelina Lamma. Abductive logic programming as an effective technology for the static verification of declarative business processes. *Fundamenta Informaticae*, 102(3):325, 2010.
- [57] Shoichi Morimoto. A Survey of Formal Verification for Business Process Modeling. In *Computational Science – ICCS 2008*, volume 5102, pages 514–522. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.
- [58] Geoffrey Muchiri Muketha, A. A. A. Ghani, M. H. Selamat, and R. Atan. A survey of business process complexity metrics. *Information Technology Journal*, 9(7):1336–1344, 2010.
- [59] Shin Nakajima. Model-Checking Behavioral Specification of BPEL Applications. *Electronic Notes in Theoretical Computer Science*, 151(2):89–105, May 2006.
- [60] Sven Overhage, Dominik Q. Birkmeier, and Sebastian Schlauderer. Quality marks, metrics, and measurement procedures for business process models. *Business & Information Systems Engineering*, 4(5):229–246, 2012.
- [61] Carl Petri. Fundamentals of a Theory of Asynchronous Information Flow. pages 386–390, 1962.
- [62] Frank Puhlmann. Soundness Verification of Business Processes Specified in the Pi-Calculus. In *On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS*, volume 4803, pages 6–23. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.
- [63] Frank Puhlmann and Mathias Weske. Investigations on Soundness Regarding Lazy Activities. In *Business Process Management*, volume 4102, pages 145–160. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

- [64] Elke Pulvermueller, Sven Feja, and Andreas Speck. Developer-friendly verification of process-based systems. *Knowledge-Based Systems*, 23(7):667–676, 2010.
- [65] Ivo Raedts, Marija Petkovic, Yaroslav S. Usenko, Jan Martijn EM van der Werf, Jan Friso Groote, and Lou J. Somers. Transformation of BPMN Models for Behaviour Analysis. In *MSVVEIS*, pages 126–137, 2007.
- [66] Mohamed Ramadan, Hicham G. Elmongui, and Riham Hassan. Bpmn formalisation using coloured petri nets. In *The 2nd GSTF Annual International Conference on Software Engineering & Applications (SEA'11)*, 2011.
- [67] Manfred Reichert and Barbara Weber. *Enabling flexibility in process-aware information systems: challenges, methods, technologies*. Springer, 2012.
- [68] Hajo A. Reijers, Jan Mendling, and Jan Recker. Business Process Quality Management. In *Handbook on Business Process Management 1*, pages 167–185. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.
- [69] Hajo A. Reijers and Irene TP Vanderfeesten. Cohesion and coupling metrics for workflow process design. In *Business Process Management*, pages 290–305. Springer, 2004.
- [70] Luis Reynoso, Elvira Rolón, Marcela Genero, Félix García, Francisco Ruiz, and Mario Piattini. Formal Definition of Measures for BPMN Models. In *Software Process and Product Measurement*, number 5891 in Lecture Notes in Computer Science, pages 285–306. Springer Berlin Heidelberg, 2009.
- [71] Elvira Rolón, Jorge Cardoso, Félix García, Francisco Ruiz, and Mario Piattini. Analysis and Validation of Control-Flow Complexity Measures with BPMN Process Models. In *Enterprise, Business-Process and Information Systems Modeling*, volume 29, pages 58–70. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.
- [72] Elvira Rolón, Francisco Ruiz, Félix García, and Mario Piattini. Applying software metrics to evaluate business process models. *CLEI-Electronic Journal*, 9(1), 2006.
- [73] Elvira Rolon, Laura Sanchez, Félix Garcia, Francisco Ruiz, Mario Piattini, Danilo Caivano, and Giuseppe Visaggio. Prediction Models for BPMN Usability and Maintainability. pages 383–390. IEEE, July 2009.
- [74] Shazia Sadiq, Guido Governatori, and Kioumars Namiri. Modeling control objectives for business process compliance. In *Business process management*, pages 149–164. Springer, 2007.
- [75] Laura Sánchez-González, Félix García, Jan Mendling, and Francisco Ruiz. Quality assessment of business process models based on thresholds. In *On the Move to Meaningful Internet Systems: OTM 2010*, pages 78–95. Springer, 2010.
- [76] Laura Sánchez-González, Félix García, Francisco Ruiz, and Jan Mendling. Quality indicators for business process models from a gateway complexity perspective. *Information and Software Technology*, 54(11):1159–1174, 2012.
- [77] Laura Sánchez-González, Félix García, Francisco Ruiz, and Mario Piattini. Toward a quality framework for business process models. *International Journal of Cooperative Information Systems*, 22(01):1350003, March 2013.
- [78] Laura Sánchez-González, Francisco Ruiz, Félix García, and Jorge Cardoso. Towards thresholds of control flow complexity measures for BPMN models. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, pages 1445–1450. ACM, 2011.

- [79] Andrew Sears. Layout appropriateness: A metric for evaluating user interface widget layout. *Software Engineering, IEEE Transactions on*, 19(7):707–719, 1993.
- [80] Franziska Semmelrodt, David Knuplesch, and Manfred Reichert. Modeling the resource perspective of business process compliance rules with the extended compliance rule graph. In *Enterprise, Business-Process and Information Systems Modeling*, pages 48–63. Springer, 2014.
- [81] Darius Silingas and Edita Mileviciene. Refactoring BPMN Models: From ‘Bad Smells’ to Best Practices and Patterns. *BPMN 2.0 Handbook Second Edition: Methods, Concepts, Case Studies and Standards in Business Process Management Notation*, page 125, 2011.
- [82] Bruce Silver. Ten tips for effective process modeling. *BPMInstitute.org.*, 9(10):2013, 2008.
- [83] Bruce Silver. *BPMN method and style: with BPMN implementer’s guide*. Cody-Cassidy Press, Aptos, Calif, 2. ed edition, 2011.
- [84] Wil MP Van Der Aalst. The Application of Petri Nets to Workflow Management. *Journal of Circuits, Systems and Computers*, 08(01):21–66, February 1998.
- [85] Wil MP van der Aalst. Workflow Verification: Finding Control-Flow Errors Using Petri-Net-Based Techniques. In *Business Process Management*, volume 1806, pages 161–183. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.
- [86] Wil MP van der Aalst and Arthur HM ter Hofstede. Verification Of Workflow Task Structures: A Petri-net-baset Approach. *Information Systems*, 25(1):43–69, March 2000.
- [87] Wil MP van der Aalst, Kees M. van Hee, Arthur HM ter Hofstede, Natalia Sidorova, H. M. W. Verbeek, Marc Voorhoeve, and Moe Thandar Wynn. Soundness of workflow nets: classification, decidability, and analysis. *Formal Aspects of Computing*, 23(3):333–363, 2011.
- [88] Boudewijn F. van Dongen, Monique H. Jansen-Vullers, H. M. W. Verbeek, and Wil MP van der Aalst. Verification of the SAP reference models using EPC reduction, state-space analysis, and invariants. *Computers in Industry*, 58(6):578–601, 2007.
- [89] Kees van Hee, Natalia Sidorova, and Marc Voorhoeve. Soundness and Separability of Workflow Nets in the Stepwise Refinement Approach. In *Applications and Theory of Petri Nets 2003*, volume 2679, pages 337–356. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.
- [90] Irene Vanderfeesten, Hajo A. Reijers, Jan Mendling, Wil MP van der Aalst, and Jorge Cardoso. On a quest for good process models: the cross-connectivity metric. In *Advanced Information Systems Engineering*, pages 480–494. Springer, 2008.
- [91] Henricus MW Verbeek, Twan Basten, and Wil MP van der Aalst. Diagnosing workflow processes using Woflan. *The computer journal*, 44(4):246–279, 2001.
- [92] Mathias Weske. *Business Process Management*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.
- [93] Peter Y. H. Wong and Jeremy Gibbons. A Process-Algebraic Approach to Workflow Specification and Refinement. In *Software Composition*, volume 4829, pages 51–65. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.
- [94] Peter Y. H. Wong and Jeremy Gibbons. A Process Semantics for BPMN. In *Formal Methods and Software Engineering*, volume 5256, pages 355–374. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.
- [95] Peter Y. H. Wong and Jeremy Gibbons. Verifying Business Process Compatibility (Short Paper). pages 126–131. IEEE, August 2008.

- [96] Moe Thandar Wynn, H. M. W. Verbeek, Wil MP van der Aalst, Arthur HM ter Hofstede, and David Edmond. Business process verification-finally a reality! *Business Process Management Journal*, 15(1):74–92, 2009.
- [97] JianHong Ye, ShiXin Sun, Lijie Wen, and Wen Song. Transformation of BPMN to YAWL. pages 354–359. IEEE, 2008.