ISCTE IUL
Instituto Universitário de Lisboa

School of Technology and Architecture

Department of Information Science and Technology

Checking and Improving Business Process Models in BPMN2

Hugo Miguel Salva Barona

A dissertation presented in partial fulfilment of the requirements for the degree of

Master in Information Systems Business Management

Supervisor:
Fernando Brito e Abreu, PhD, Associate Professor, DCTI/ISCTE-IUL

December, 2015

Abstract

Business Process Modeling (BPM) is a systems engineering activity where we represent the processes of an enterprise, so they can be shared, understood and improved. Despite the set of innovative tools for BPM modelling that exist in the market, they allow modelers to introduce errors during the modelling process. As there is no idea which errors the tools do not detect, what are the most recurrent errors and how could this problem be mitigated, this dissertation presents a study and a proposal to help solving this problem. Firstly, a tool survey was developed to describe the state of the practice on the ability of Modelling Tools to validate BPMN2 models and determine the most recurrent defects introduced by BPMN modellers. Secondly, based on an empirical study using the QUASAR validator we provide evidence on its ability to validate a set of well-formedness rules and best practices and therefore detect errors in BPMN2 Models. Finally, we want to understand if this metamodelling-based validation facility can be used to prevent introducing modelling errors, while speeding up the learning curve.

Keywords: *BPMN models validation; business process models validation; business process modeling error prevention.*

[This page was intentionally left blank]

Resumo

A Modelação de Processos de Negócio (MPN) é uma atividade de engenharia de sistemas onde representamos os processos de uma empresa, para que os mesmos possam ser partilhados, compreendidos e melhorados. Apesar do elevado número de ferramentas de MPN existentes no mercado, estas permitem aos modeladores introduzir erros du- rante o processo de modelação. Como não existe uma ideia clara acerca de quais os erros que as ferramentas não detetam, quais os erros cometidos mais recorrentemente e como o problema pode ser resolvido, esta dissertação apresenta um estudo e uma pro- posta para resolver o problema. Inicialmente foi efetuado um levantamento do estado da prática da capacidade das ferramentas de modelação para validar os modelos em BPMN2, e determinar os erros mais frequentemente introduzidos pelos modeladores. Em seguida, baseado num estudo empírico, usando o validador QUASAR, fornecemos evidências sobre a sua capacidade para validar o conjunto de regras de boa formação e boas práticas na modelação de processos de negócio e assim detetar os erros introduzidos nos modelos em BPMN2. Finalmente, queremos compreender se esta facilidade de validação baseada em metamodelos pode ser usada para prevenir a introdução de erros durante o processo de modelação de processos de negócio, acelerando assim a curva de aprendizagem do modelador.

Palavras-chave: *validação modelos BPMN; validação de modelos de processos de negócio; prevenção na modelação de processos de negócio.*

[This page was intentionally left blank]

1. INTRODUCTION

1.1 Scope

Business process modeling (BPM) allows the user to represent business processes, so they may be analysed and/or improved. Often, the main goal is to identify cost-intensive activities, increase speed and/or quality, to reduce cycle time or costs. Several business process modeling techniques are available such as flowcharts, data flows diagrams, gantt charts, Petri nets, UML activity diagrams and much more. However, there has been an increasing trend (see figures 1.1 and 1.2) for using BPMN (Business Process Modeling Notation) for modeling business processes, as well as in other knowledge areas, such as modeling software development processes [5] and [6], web service applications [7] or ETL processes [8]. The BPMN modeling notation can be defined as a set of graphical elements and rules defining connections between them. Existing BPMN modeling tools allow the user to model their business processes, validate and simulate the execution of those models. Those functionalities are expected to allow the user to better improve and optimize its BPM models.

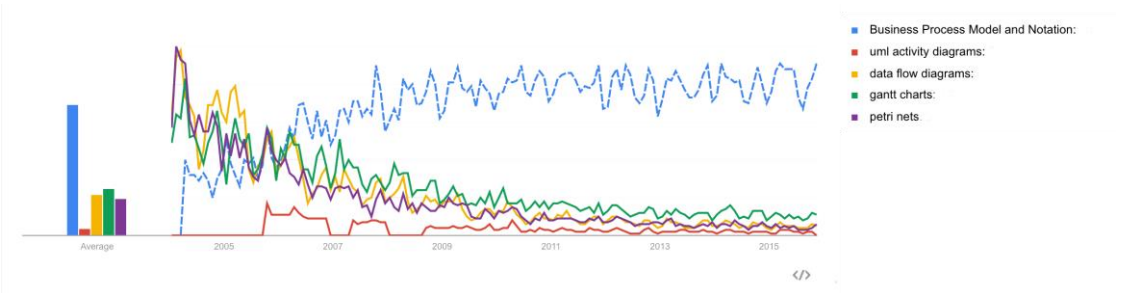


Fig. 1.1: BPMN Google Search Interest relatively other notations from 2004 to 2015.

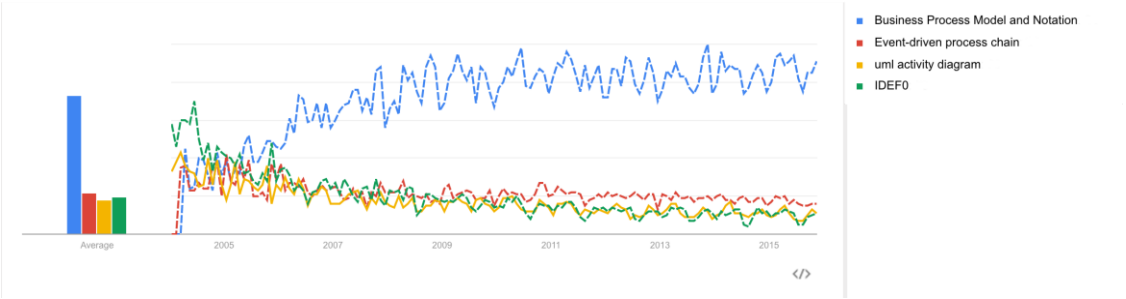


Fig. 1.2: BPMN Google Search Interest relatively other notations from 2004 to 2015.

BPMN is represented by syntax and semantic rules. Syntax and semantics of graphical notations can be expressed by well-formedness rules defined upon the corresponding language metamodel. That has been a common practice for OMG's (Object Managament Group) metamodels, which uses a combination of UML (Unified Modelling Language) and OCL (Object Constraints Language) invariants on that matter [9]. However, the official BPMN2 (Business Process Modeling and Notation – version 2) metamodel only presents those rules in natural language [10]. At the QUASAR (Quantitative Approaches in Software Engineering and Reengineering) research group and in the scope of Anacleto Correia PhD thesis work [11], was produced a formalization of both well-formedness rules and best practices using OCL [12]. The resulting BPMN2 metamodel was used to produce a validation mechanism for BPMN2 models, based on the USE (UML-based Specification Environment) environment, accessed with the J-USE API (Application Program Interface in Java). This initiative was the groundwork that allowed us to elicit the set of research questions to be presented ahead.

1.2 Motivation

Business process modeling is an important instrument for business process management, since it may allow that relevant stakeholders, such as managers, analysts, users and information system designers to participate, and together, to improve an organization's business processes [13]. Business process models are expressed in a given process modeling language and, although it exists for many years now, they were not deemed to be easily used and understood by business and IT parties in order to communicate relevant business process semantics. As so, the OMG commissioned the development of the BPMN, currently standing as version 2. BPMN2 has become a de facto standard in industry and services. According to the OMG, it may be usable and understandable for all busi- ness users, from the business analysts that create the initial drafts of the processes, to the technical developers responsible for implementing the technology that will perform those processes, and finally to the business people who will manage and monitor those

processes [10].

The effectiveness of a model, as a shared understanding of the problem domain, may depend on two points of analysis: language expressiveness and model validity. In this research work we will be concerned with the latter. Model validity concerns syntax and semantic aspects. BPMN2 is a syntactically rich modeling language. While, for instance, a UML activity diagram has around 20 different modeling constructs, a BPMN2 process model diagram has around 100 different modeling constructs, including 51 events types, 8 gateway types, 7 data types, 4 types of activities, 6 activity markers, 7 task types, 4 flow types, pools, lanes, etc. If BPMN2 modellers are given the freedom to combine such a large plethora of modeling constructs, in the absence of a powerful validation or recommendation mechanism, embedded in the modeling tool, it may arise inconsistencies in designed models [12].

Regarding models semantics, the mix of constructs found in BPMN2 might make it possible to obtain models with a wide range of semantic errors that make BPMN models overly complex, difficult to understand and maintain. According to [14], there might be a lack of discussion on *bad smells* in BPMN models. Several authors have tried to express sets of BPMN modeling best practices to improve model semantics such as [15], [16] and [17]. However, we are not yet aware of any facility to streamline those practices in the modeling activity. Hereinafter, we will use the term ”model smell” as a synonym to a violation of one such best practice or to a well-formedness rule defined in the BPMN2 standard. In other words, we can have as many BPMN2 model smells as the cardinality of the union of best practices and well-formedness rules.

On the other side, the known studies related with comparison of BPM tools, are few, or are very focused on a specific in a tool characteristic [18], or are very wide but don’t consider to analysis the validation capacity of BPM tools [19].

1.3 Context

For illustration purposes, we describe three model smells examples, two related with well-formedness rules and one related with best practices.

1.3.1 Outgoing Sequence Flow not allowed in an End Event.

The end event indicates where a process will end. In terms of sequence flows, the end event ends the flow of the process, and thus, will not have any outgoing sequence flows.

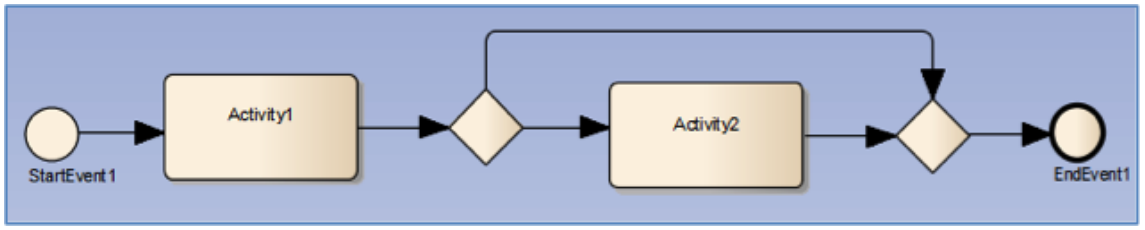


Fig. 1.3: **Correct:** End Event has no outgoing sequence flows.

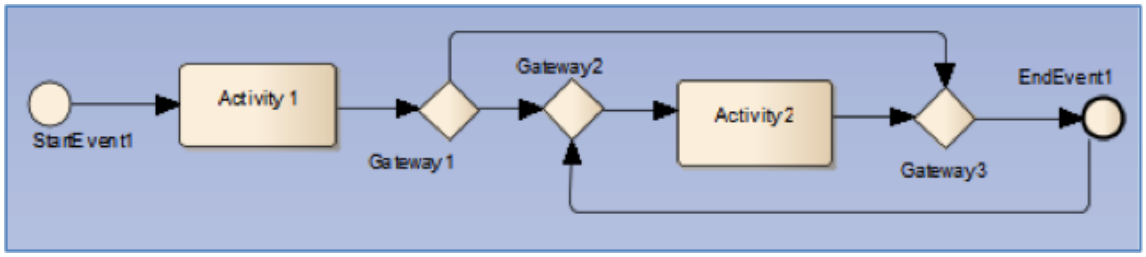


Fig. 1.4: **Wrong:** End Event has an outgoing sequence flow.

1.3.2Catch Error Event must trigger an exception flow

Error intermediate events cannot be used within normal sequence flows. This catching event should trigger an exception flow.

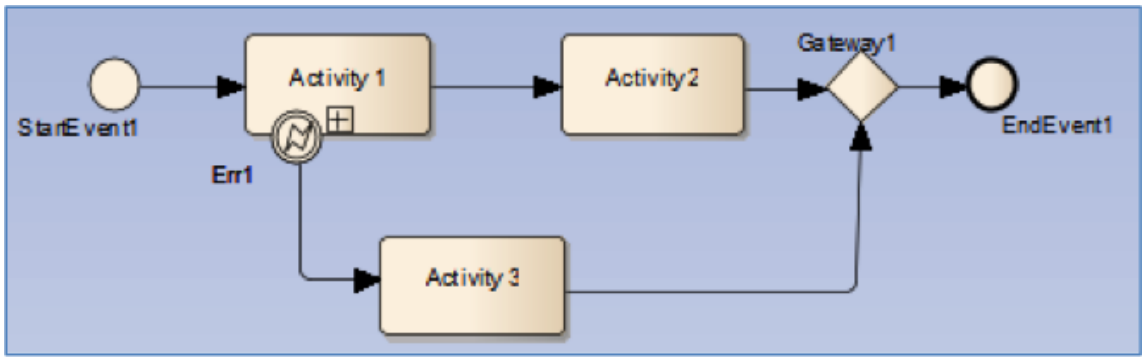


Fig. 1.5: **Correct:** Error Event triggering an exception flow.

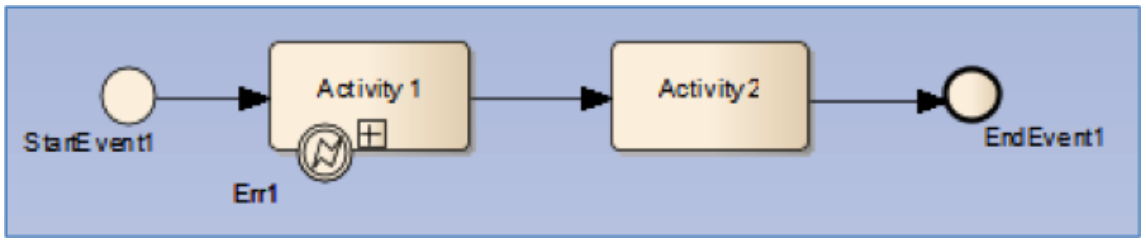


Fig. 1.6: **Wrong:** Error Event without an exception flow.

1.3.3 Use explicitly Start Events and End Events

Process modeling best practices recommendations advise the explicit use of start and end events.

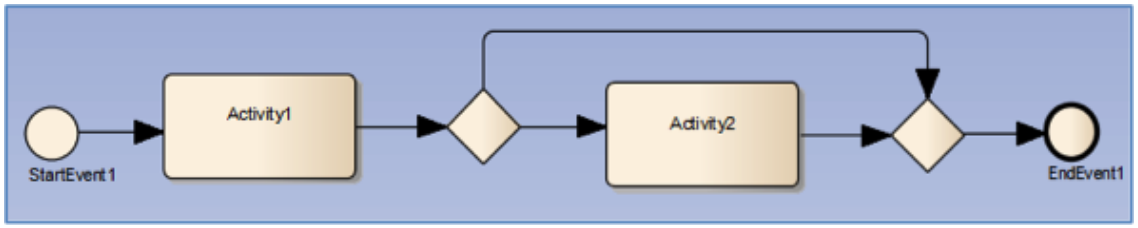


Fig. 1.7: **Use:** Explicit of Start and End Events.

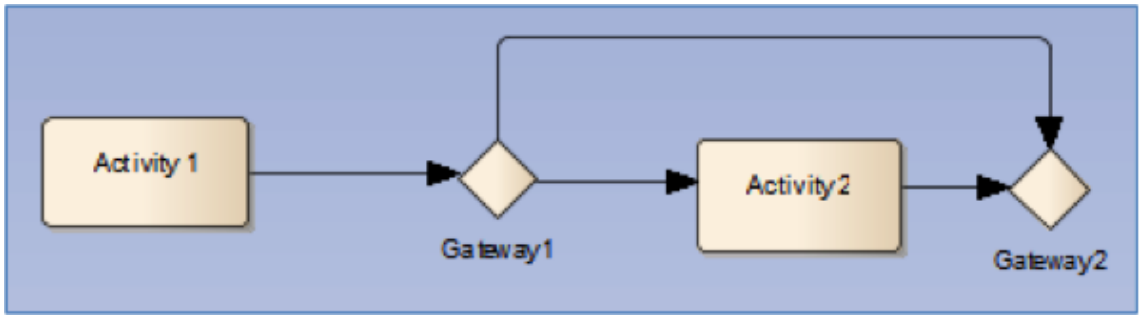


Fig. 1.8: **Avoid:** Use of Flow Nodes as implicit Start and End Events.

1.4 Research problems and expected contributions

1.4.1 Objective

The purpose of this dissertation is: first we want to characterize in detail the current state of practice regarding the detection of design errors in BPMN2 modeling tools and, second, to assess how a metamodeling facility can be used to improve such state of practice. To reach the first objective a tool survey was carried out and is presented in this dissertation, where the set of well-formedness rules and best practices rules defined in [12] were used in order to validate if they are checked by a set of BPMN2 modeling tools. To fulfill the second objective, we will integrate a metamodel-based validation facility, developed in the QUASAR group, in an open-source BPMN2 tool and perform an empirical study to check if such facility speeds up BPMN2 learning curve and reduce the resulting model defects.

1.4.2 Contribution

We identified the following contributions in our work:

- Description of the current state of practice in model validation facilities provided by existing BPMN modeling tools.
- Characterization of the distribution of BPMN modeling smells (defects).
- Finding out if some model defects are less prone for detection than others.
- Finding out if a significant improvement in the process of validating BPMN models can be obtained by applying our metamodel-based approach.
- Finding out if the learning curve in BPMN modeling can be flattened by using an appropriate recommendation facility.

1.4.3 Research Questions

1. RQ1: Do current state-of-practice BPMN2 modeling tools detect the violation of well-formedness rules, as defined in OMG's metamodel?
2. RQ2: Do current state-of-practice BPMN2 modeling tools detect the violation of known best practices?
3. RQ3: Can a metamodeling-based validation facility be used to prevent introducing modeling errors?
4. RQ4: Can a metamodeling-based validation facility be used to speed up the learning curve?

1.4.4 Research Methodology

The methodology that we followed encompassed the following steps:

1. **Become proficient in BPMN2 modeling, namely on well-formedness rules**
   (a) Objective: obtain background knowledge to understand the problem domain
2. **Perform an empirical survey of BPMN2 validation capabilities (blackbox) on current state-of-the-practice tools (see section 2.1). This survey will encompass data collection based upon a set of BPMN modeling smells identified and represented as BPMN model snippets in [20]**
   (a) Objective: answer RQ1 and RQ2
3. **Integrate the metamodel-based QUASAR validation mechanism for BPMN2 models to an open source BPMN modeling tool.**

(a) Objective: build a prototype environment to enable data collection for the last step

4. **Perform an experiment on the usage of the prototype to evaluate its effectiveness in improving the business process modeling activity. A set of group works developed by students in academy is going to be used to support the experiment.**

   (a) Objective: answer RQ3 and RQ4

[This page was intentionally left blank]

2. SYSTEMATIC BPMN TOOL ASSESSMENT

We have performed a systematic assessment of BPMN2 process modeling tools in order to identify possible failures in validation of business process models. The analysis will consist of design a model smell that do not respect the rule for each of the well-formedness rules and best practices in each of the tools considered, and validate if the tool would detect the validation defect or not. During this process we will collect information taking into consideration the tool used and the well-formedness rule or best practice analysed, and thereafter analyse the data and provide some conclusions and answers that could be of value for our questions. In terms of defects, it was considered to analysis a total of one hundred and twenty-two defects (122), which thirty (30) are defects in best practices rules category and ninety-two (92) are defects in well-formedness rules category.

2.1 Tools Selection Process

Many BPMN tools exist nowadays. As early as in 2010, *Gartner* performed a comparative assessment of those tools (figure 2.2). As observed previously, *BPMN* has attained a large popularity and is currently the most widely used process modeling notation, and even increased number of BPMN tools, either proprietary, freeware or open source have emerged in recent years, as reported by Wikipedia (https://en.wikipedia.org/wiki/Comparison_of_Business_Process_Modeling_Notation_tools).

Due to time constraints we could not assess all tools, so we have performed a mix of stratified and convenience sampling. The strata were (1) open source, (2) freeware tools and (3) proprietary tools. Within each strata we used convenience sampling based on the availability and installation ability. Indeed, some open source tools could not be installed and some proprietary tools had no fully functional demo / trial versions for us to perform our assessment exercise. The final list of tools that met our and availability criteria are represented in table 2.1.

Table 2.1: BPMN2 Tools Final Selection

Tool Name	Creator	Version	Licesing Model
Aris Express	Software AG	2.4	Freeware
Bizagi Process Modeler	Bizagi	2.8	Freeware
Bonita BPM	Bonitasoft	7.0	Open Source
Eclipse BPMN2 Modeler	Eclipse	1.0	Open Source
Enterprise Architect	Sparx Systems	11.1	Proprietary
Intalio Bpms	Intalio	N/A	Freeware
jBPM	Jboss project	6.1.0	Open Source
Modelio	Modeliosoft	3.1.2	Open Source
Process Modeler for Microsoft Visio	Itp commerce AG	5	Proprietary
Tibco Business Studio - BPM Edition	TIBCO Software Inc.	3.9.0	Proprietary
Yaoqiang BPMN Editor	Blenta (Sourceforge ID)	2.2.3	Open Source

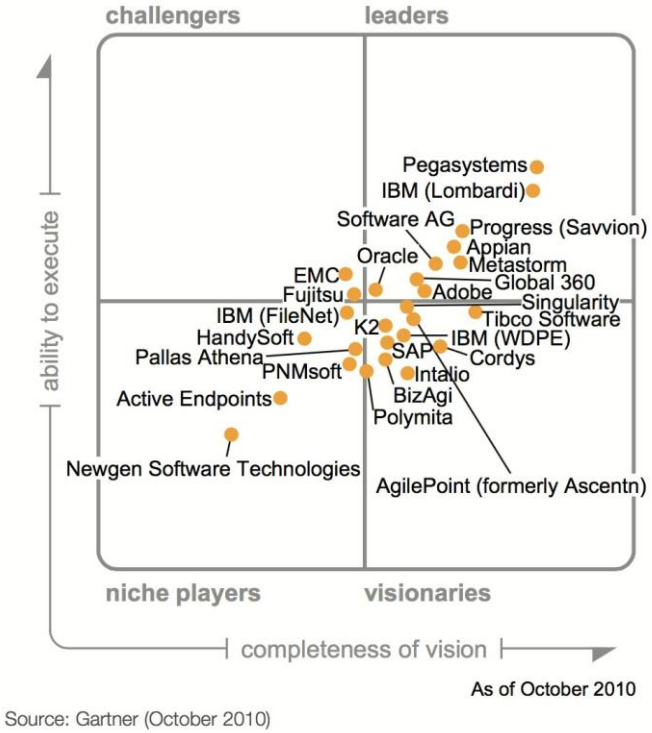


Fig. 2.1: Gartner Magic Quadrant for BPMN Suite Tools [1]

[1](source: Adobe - [https://goo.gl/f48SJR])

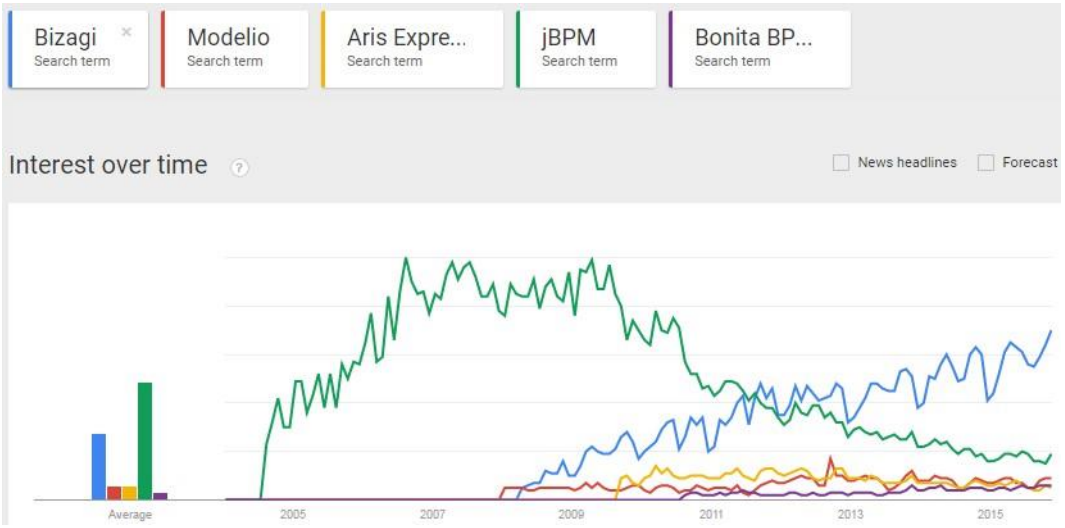


Fig. 2.2: Bizagi Google Search Interest relatively other modeling tools from 2005 to 2015. [2]

2.2 Data Collection

To assess the validation capabilities of each selected tool, data was collected for each tool and for each well-formedness rule or best practice. Basically the information collected indicates if the specific tool is able to identify the model smell tested. The figure 2.3 represents the distribution of model smells considered in this analysis for each category of BPMN2 elements.

		Category		
		Best Practice	Well-formedness	Total
Scope	Activity	5	6	11
	Artifacts	0	3	3
	Collaboration	0	1	1
	Data Flow	0	4	4
	Events	13	47	60
	Flow Nodes	0	1	1
	Gateway	10	15	25
	Message Flow	1	4	5
	Process	1	4	5
	Sequence Flow	0	3	3
	Sub-Process	0	4	4
Total		30	92	122

Fig. 2.3: Model Smells Distribution

[2](source: [https://www.google.com.gi/trends/])

2.2.1Existing Limitations

Based on the existing limitations of the BPM tools in terms of BPMN2 support, some rules from the set of rules considered to this analysis were not possible to validate in the tools, since these tools do not support them. To give a practical example of this limitation, in the next table 2.2 we have two invariants expressed that were not possible to analyse in Bizagi tool, since the tool does not support them. Both invariants belong to *Gateway* scope, and are related with Gateway direction. The BPMN2 specification [10] defines that a Gateway must have a direction defined, and this direction can only be one of two options : converging or diverging. The first invariant with name *converging* validates if all Gateway elements, with converging direction defined, do not have more than one outgoing sequence flow, and have more than one incoming sequence flow. The second invariant with name *diverging* validates if all Gateway elements, with diverging direction defined, do not have more than one incoming sequence flow, and have more that one outgoing sequence flow. On Bizagi tool, since there is no property in Gateway element related with direction, it is not possible to specify a direction for Gateways. Therefore, the tool does not support the validation of these invariants.

Table 2.2: Ivariant examples without support on Bizagi tool.

Invariant ID	Invariant Scope	Invariant Name
69	Gateway	converging
70	Gateway	diverging

The full list related with tools coverage for model smells considered into analysis is available as appendix - figure B.1.

2.3 Data Analysis

We have performed the following analysis using the IBM SPSS Statistics Software tool. Several graphs were produced to support our analysis.

2.3.1 Model Smells Detection per tool

The next sequence of figures from 2.4 to 2.14 represent the analysis of detection of model smells in detail for each scope, where each figure represents the results of each tool considered for the analysis.

The next figure 2.4 doesn't has represented the column for Detection equals one (1) for best practice rules category, since there is no scope with detection for this category of rules, consequently the column was omitted in order to facilitate the read of table's information.

			Well-formedness Rules			Best Practice Rules		All Rule Types		
			Detection			Detection		Detection		
			0	1	Total	0	Total	0	1	Total
Scope	Activity	Count	6	0	6	5	5	11	0	11
		% within Detection	100.0%	0.0%	100.0%	100.0%	100.0%	100.0%	0.0%	100.0%
	Artifacts	Count	0	2	2	0	0	0	2	2
		% within Detection	0.0%	100.0%	100.0%	0.0%	0.0%	0.0%	100.0%	100.0%
	Collaboration	Count	1	0	1	0	0	1	0	1
		% within Detection	100.0%	0.0%	100.0%	0.0%	0.0%	100.0%	0.0%	100.0%
	Data Flow	Count	3	0	3	0	0	3	0	3
		% within Detection	100.0%	0.0%	100.0%	0.0%	0.0%	100.0%	0.0%	100.0%
	Events	Count	40	7	47	13	13	53	7	60
		% within Detection	85.1%	14.9%	100.0%	100.0%	100.0%	88.3%	11.7%	100.0%
	Flow Nodes	Count	1	0	1	0	0	1	0	1
		% within Detection	100.0%	0.0%	100.0%	0.0%	0.0%	100.0%	0.0%	100.0%
	Gateway	Count	12	0	12	10	10	22	0	22
		% within Detection	100.0%	0.0%	100.0%	100.0%	100.0%	100.0%	0.0%	100.0%
	Message Flow	Count	4	0	4	1	1	5	0	5
		% within Detection	100.0%	0.0%	100.0%	100.0%	100.0%	100.0%	0.0%	100.0%
	Process	Count	0	2	2	1	1	1	2	3
		% within Detection	0.0%	100.0%	100.0%	100.0%	100.0%	33.3%	66.7%	100.0%
	Sequence Flow	Count	3	0	3	0	0	3	0	3
		% within Detection	100.0%	0.0%	100.0%	0.0%	0.0%	100.0%	0.0%	100.0%
	Sub-Process	Count	2	2	4	0	0	2	2	4
		% within Detection	50.0%	50.0%	100.0%	0.0%	0.0%	50.0%	50.0%	100.0%
Total		Count	72	13	85	30	30	102	13	115
		% within Detection	84.7%	15.3%	100.0%	100.0%	100.0%	88.7%	11.3%	100.0%

Fig. 2.4: Defects detection analysis on Aris Express.

			Well-formedness Rules			Best Practice Rules			All Rule Types		
			Detection			Detection			Detection		
			0	1	Total	0	1	Total	0	1	Total
Scope	Activity	Count	0	1	1	5	0	5	5	1	6
		% within Detection	0.0%	100.0%	100.0%	100.0%	0.0%	100.0%	83.3%	16.7%	100.0%
	Artifacts	Count	1	2	3	0	0	0	1	2	3
		% within Detection	33.3%	66.7%	100.0%	0.0%	0.0%	0.0%	33.3%	66.7%	100.0%
	Collaboration	Count	0	1	1	0	0	0	0	1	1
		% within Detection	0.0%	100.0%	100.0%	0.0%	0.0%	0.0%	0.0%	100.0%	100.0%
	Data Flow	Count	2	0	2	0	0	0	2	0	2
		% within Detection	100.0%	0.0%	100.0%	0.0%	0.0%	0.0%	100.0%	0.0%	100.0%
	Events	Count	25	9	34	11	0	11	36	9	45
		% within Detection	73.5%	26.5%	100.0%	100.0%	0.0%	100.0%	80.0%	20.0%	100.0%
	Flow Nodes	Count	1	0	1	0	0	0	1	0	1
		% within Detection	100.0%	0.0%	100.0%	0.0%	0.0%	0.0%	100.0%	0.0%	100.0%
	Gateway	Count	6	5	11	8	2	10	14	7	21
		% within Detection	54.5%	45.5%	100.0%	80.0%	20.0%	100.0%	66.7%	33.3%	100.0%
	Message Flow	Count	3	1	4	1	0	1	4	1	5
		% within Detection	75.0%	25.0%	100.0%	100.0%	0.0%	100.0%	80.0%	20.0%	100.0%
	Process	Count	1	1	2	1	0	1	2	1	3
		% within Detection	50.0%	50.0%	100.0%	100.0%	0.0%	100.0%	66.7%	33.3%	100.0%
	Sequence Flow	Count	0	3	3	0	0	0	0	3	3
		% within Detection	0.0%	100.0%	100.0%	0.0%	0.0%	0.0%	0.0%	100.0%	100.0%
	Sub-Process	Count	1	0	1	0	0	0	1	0	1
		% within Detection	100.0%	0.0%	100.0%	0.0%	0.0%	0.0%	100.0%	0.0%	100.0%
Total		Count	40	23	63	26	2	28	66	25	91
		% within Detection	63.5%	36.5%	100.0%	92.9%	7.1%	100.0%	72.5%	27.5%	100.0%

Fig. 2.5: Defects detection analysis on Intalio Designer.

			Well-formedness Rules			Best Practice Rules			All Rule Types		
			Detection			Detection			Detection		
			0	1	Total	0	1	Total	0	1	Total
Scope	Activity	Count	6	0	6	4	1	5	10	1	11
		% within Detection	100.0%	0.0%	100.0%	80.0%	20.0%	100.0%	90.9%	9.1%	100.0%
	Artifacts	Count	1	2	3	0	0	0	1	2	3
		% within Detection	33.3%	66.7%	100.0%	0.0%	0.0%	0.0%	33.3%	66.7%	100.0%
	Collaboration	Count	0	1	1	0	0	0	0	1	1
		% within Detection	0.0%	100.0%	100.0%	0.0%	0.0%	0.0%	0.0%	100.0%	100.0%
	Data Flow	Count	4	0	4	0	0	0	4	0	4
		% within Detection	100.0%	0.0%	100.0%	0.0%	0.0%	0.0%	100.0%	0.0%	100.0%
	Events	Count	33	13	46	12	1	13	45	14	59
		% within Detection	71.7%	28.3%	100.0%	92.3%	7.7%	100.0%	76.3%	23.7%	100.0%
	Flow Nodes	Count	0	1	1	0	0	0	0	1	1
		% within Detection	0.0%	100.0%	100.0%	0.0%	0.0%	0.0%	0.0%	100.0%	100.0%
	Gateway	Count	14	0	14	8	1	9	22	1	23
		% within Detection	100.0%	0.0%	100.0%	88.9%	11.1%	100.0%	95.7%	4.3%	100.0%
	Message Flow	Count	4	0	4	1	0	1	5	0	5
		% within Detection	100.0%	0.0%	100.0%	100.0%	0.0%	100.0%	100.0%	0.0%	100.0%
	Process	Count	2	1	3	1	0	1	3	1	4
		% within Detection	66.7%	33.3%	100.0%	100.0%	0.0%	100.0%	75.0%	25.0%	100.0%
	Sequence Flow	Count	1	2	3	0	0	0	1	2	3
		% within Detection	33.3%	66.7%	100.0%	0.0%	0.0%	0.0%	33.3%	66.7%	100.0%
	Sub-Process	Count	3	1	4	0	0	0	3	1	4
		% within Detection	75.0%	25.0%	100.0%	0.0%	0.0%	0.0%	75.0%	25.0%	100.0%
Total		Count	68	21	89	26	3	29	94	24	118
		% within Detection	76.4%	23.6%	100.0%	89.7%	10.3%	100.0%	79.7%	20.3%	100.0%

Fig. 2.6: Defects detection analysis on Eclipse BPMN2 Modeler.

			Well-formedness Rules			Best Practice Rules			All Rule Types		
			Detection			Detection			Detection		
			0	1	Total	0	1	Total	0	1	Total
Scope	Activity	Count	3	3	6	5	0	5	8	3	11
		% within Detection	50.0%	50.0%	100.0%	100.0%	0.0%	100.0%	72.7%	27.3%	100.0%
	Artifacts	Count	1	2	3	0	0	0	1	2	3
		% within Detection	33.3%	66.7%	100.0%	0.0%	0.0%	0.0%	33.3%	66.7%	100.0%
	Collaboration	Count	0	1	1	0	0	0	0	1	1
		% within Detection	0.0%	100.0%	100.0%	0.0%	0.0%	0.0%	0.0%	100.0%	100.0%
	Data Flow	Count	4	0	4	0	0	0	4	0	4
		% within Detection	100.0%	0.0%	100.0%	0.0%	0.0%	0.0%	100.0%	0.0%	100.0%
	Events	Count	38	8	46	12	1	13	50	9	59
		% within Detection	82.6%	17.4%	100.0%	92.3%	7.7%	100.0%	84.7%	15.3%	100.0%
	Flow Nodes	Count	1	0	1	0	0	0	1	0	1
		% within Detection	100.0%	0.0%	100.0%	0.0%	0.0%	0.0%	100.0%	0.0%	100.0%
	Gateway	Count	13	0	13	10	0	10	23	0	23
		% within Detection	100.0%	0.0%	100.0%	100.0%	0.0%	100.0%	100.0%	0.0%	100.0%
	Message Flow	Count	4	0	4	1	0	1	5	0	5
		% within Detection	100.0%	0.0%	100.0%	100.0%	0.0%	100.0%	100.0%	0.0%	100.0%
	Process	Count	1	1	2	1	0	1	2	1	3
		% within Detection	50.0%	50.0%	100.0%	100.0%	0.0%	100.0%	66.7%	33.3%	100.0%
	Sequence Flow	Count	2	1	3	0	0	0	2	1	3
		% within Detection	66.7%	33.3%	100.0%	0.0%	0.0%	0.0%	66.7%	33.3%	100.0%
	Sub-Process	Count	1	3	4	0	0	0	1	3	4
		% within Detection	25.0%	75.0%	100.0%	0.0%	0.0%	0.0%	25.0%	75.0%	100.0%
Total		Count	68	19	87	29	1	30	97	20	117
		% within Detection	78.2%	21.8%	100.0%	96.7%	3.3%	100.0%	82.9%	17.1%	100.0%

Fig. 2.7: Defects detection analysis on Bizagi Process Modeler.

			Well-formedness Rules			Best Practice Rules			All Rule Types		
			Detection			Detection			Detection		
			0	1	Total	0	1	Total	0	1	Total
Scope	Activity	Count	5	1	6	4	1	5	9	2	11
		% within Detection	83.3%	16.7%	100.0%	80.0%	20.0%	100.0%	81.8%	18.2%	100.0%
	Artifacts	Count	1	2	3	0	0	0	1	2	3
		% within Detection	33.3%	66.7%	100.0%	0.0%	0.0%	0.0%	33.3%	66.7%	100.0%
	Collaboration	Count	0	1	1	0	0	0	0	1	1
		% within Detection	0.0%	100.0%	100.0%	0.0%	0.0%	0.0%	0.0%	100.0%	100.0%
	Data Flow	Count	4	0	4	0	0	0	4	0	4
		% within Detection	100.0%	0.0%	100.0%	0.0%	0.0%	0.0%	100.0%	0.0%	100.0%
	Events	Count	24	20	44	11	2	13	35	22	57
		% within Detection	54.5%	45.5%	100.0%	84.6%	15.4%	100.0%	61.4%	38.6%	100.0%
	Flow Nodes	Count	0	0	0	0	0	0	0	0	0
		% within Detection	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
	Gateway	Count	9	4	13	9	1	10	18	5	23
		% within Detection	69.2%	30.8%	100.0%	90.0%	10.0%	100.0%	78.3%	21.7%	100.0%
	Message Flow	Count	2	2	4	1	0	1	3	2	5
		% within Detection	50.0%	50.0%	100.0%	100.0%	0.0%	100.0%	60.0%	40.0%	100.0%
	Process	Count	2	1	3	1	0	1	3	1	4
		% within Detection	66.7%	33.3%	100.0%	100.0%	0.0%	100.0%	75.0%	25.0%	100.0%
	Sequence Flow	Count	2	1	3	0	0	0	2	1	3
		% within Detection	66.7%	33.3%	100.0%	0.0%	0.0%	0.0%	66.7%	33.3%	100.0%
	Sub-Process	Count	4	0	4	0	0	0	4	0	4
		% within Detection	100.0%	0.0%	100.0%	0.0%	0.0%	0.0%	100.0%	0.0%	100.0%
Total		Count	53	32	85	26	4	30	79	36	115
		% within Detection	62.4%	37.6%	100.0%	86.7%	13.3%	100.0%	68.7%	31.3%	100.0%

Fig. 2.8: Defects detection analysis on Tibco Business Studio.

			Well-formedness Rules			Best Practice Rules			All Rule Types		
			Detection			Detection			Detection		
			0	1	Total	0	1	Total	0	1	Total
Scope	Activity	Count	1	3	4	1	4	5	2	7	9
		% within Detection	25.0%	75.0%	100.0%	20.0%	80.0%	100.0%	22.2%	77.8%	100.0%
	Artifacts	Count	1	2	3	0	0	0	1	2	3
		% within Detection	33.3%	66.7%	100.0%	0.0%	0.0%	0.0%	33.3%	66.7%	100.0%
	Collaboration	Count	0	1	1	0	0	0	0	1	1
		% within Detection	0.0%	100.0%	100.0%	0.0%	0.0%	0.0%	0.0%	100.0%	100.0%
	Data Flow	Count	0	0	0	0	0	0	0	0	0
		% within Detection	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
	Events	Count	13	20	33	6	6	12	19	26	45
		% within Detection	39.4%	60.6%	100.0%	50.0%	50.0%	100.0%	42.2%	57.8%	100.0%
	Flow Nodes	Count	1	0	1	0	0	0	1	0	1
		% within Detection	100.0%	0.0%	100.0%	0.0%	0.0%	0.0%	100.0%	0.0%	100.0%
	Gateway	Count	5	2	7	8	2	10	13	4	17
		% within Detection	71.4%	28.6%	100.0%	80.0%	20.0%	100.0%	76.5%	23.5%	100.0%
	Message Flow	Count	1	3	4	0	1	1	1	4	5
		% within Detection	25.0%	75.0%	100.0%	0.0%	100.0%	100.0%	20.0%	80.0%	100.0%
	Process	Count	2	1	3	1	0	1	3	1	4
		% within Detection	66.7%	33.3%	100.0%	100.0%	0.0%	100.0%	75.0%	25.0%	100.0%
	Sequence Flow	Count	1	2	3	0	0	0	1	2	3
		% within Detection	33.3%	66.7%	100.0%	0.0%	0.0%	0.0%	33.3%	66.7%	100.0%
	Sub-Process	Count	1	3	4	0	0	0	1	3	4
		% within Detection	25.0%	75.0%	100.0%	0.0%	0.0%	0.0%	25.0%	75.0%	100.0%
Total		Count	26	37	63	16	13	29	42	50	92
		% within Detection	41.3%	58.7%	100.0%	55.2%	44.8%	100.0%	45.7%	54.3%	100.0%

Fig. 2.9: Defects detection analysis on Bonita BPM.

The next figure 2.10 doesn't has represented the column for Detection equals one (1) for best practice rules category, since there is no scope with detection for this category of rules, consequently the column was omitted in order to facilitate the read of table's information.

			Well-formedness Rules			Best Practice Rules		All Rule Types		
			Detection			Detection		Detection		
			0	1	Total	0	Total	0	1	Total
Scope	Activity	Count	6	0	6	5	5	11	0	11
		% within Detection	100.0%	0.0%	100.0%	100.0%	100.0%	100.0%	0.0%	100.0%
	Artifacts	Count	3	0	3	0	0	3	0	3
		% within Detection	100.0%	0.0%	100.0%	0.0%	0.0%	100.0%	0.0%	100.0%
	Collaboration	Count	1	0	1	0	0	1	0	1
		% within Detection	100.0%	0.0%	100.0%	0.0%	0.0%	100.0%	0.0%	100.0%
	Data Flow	Count	3	0	3	0	0	3	0	3
		% within Detection	100.0%	0.0%	100.0%	0.0%	0.0%	100.0%	0.0%	100.0%
	Events	Count	36	6	42	13	13	49	6	55
		% within Detection	85.7%	14.3%	100.0%	100.0%	100.0%	89.1%	10.9%	100.0%
	Flow Nodes	Count	1	0	1	0	0	1	0	1
		% within Detection	100.0%	0.0%	100.0%	0.0%	0.0%	100.0%	0.0%	100.0%
	Gateway	Count	14	0	14	10	10	24	0	24
		% within Detection	100.0%	0.0%	100.0%	100.0%	100.0%	100.0%	0.0%	100.0%
	Message Flow	Count	4	0	4	1	1	5	0	5
		% within Detection	100.0%	0.0%	100.0%	100.0%	100.0%	100.0%	0.0%	100.0%
	Process	Count	4	0	4	1	1	5	0	5
		% within Detection	100.0%	0.0%	100.0%	100.0%	100.0%	100.0%	0.0%	100.0%
	Sequence Flow	Count	1	2	3	0	0	1	2	3
		% within Detection	33.3%	66.7%	100.0%	0.0%	0.0%	33.3%	66.7%	100.0%
	Sub-Process	Count	3	1	4	0	0	3	1	4
		% within Detection	75.0%	25.0%	100.0%	0.0%	0.0%	75.0%	25.0%	100.0%
Total		Count	76	9	85	30	30	106	9	115
		% within Detection	89.4%	10.6%	100.0%	100.0%	100.0%	92.2%	7.8%	100.0%

Fig. 2.10: Defects detection analysis on Enterprise Architect.

			Well-formedness Rules			Best Practice Rules			All Rule Types		
			Detection			Detection			Detection		
			0	1	Total	0	1	Total	0	1	Total
Scope	Activity	Count	4	2	6	4	1	5	8	3	11
		% within Detection	66.7%	33.3%	100.0%	80.0%	20.0%	100.0%	72.7%	27.3%	100.0%
	Artifacts	Count	0	3	3	0	0	0	0	3	3
		% within Detection	0.0%	100.0%	100.0%	0.0%	0.0%	0.0%	0.0%	100.0%	100.0%
	Collaboration	Count	1	0	1	0	0	0	1	0	1
		% within Detection	100.0%	0.0%	100.0%	0.0%	0.0%	0.0%	100.0%	0.0%	100.0%
	Data Flow	Count	3	1	4	0	0	0	3	1	4
		% within Detection	75.0%	25.0%	100.0%	0.0%	0.0%	0.0%	75.0%	25.0%	100.0%
	Events	Count	19	26	45	12	1	13	31	27	58
		% within Detection	42.2%	57.8%	100.0%	92.3%	7.7%	100.0%	53.4%	46.6%	100.0%
	Flow Nodes	Count	0	1	1	0	0	0	0	1	1
		% within Detection	0.0%	100.0%	100.0%	0.0%	0.0%	0.0%	0.0%	100.0%	100.0%
	Gateway	Count	5	9	14	10	0	10	15	9	24
		% within Detection	35.7%	64.3%	100.0%	100.0%	0.0%	100.0%	62.5%	37.5%	100.0%
	Message Flow	Count	4	0	4	1	0	1	5	0	5
		% within Detection	100.0%	0.0%	100.0%	100.0%	0.0%	100.0%	100.0%	0.0%	100.0%
	Process	Count	3	0	3	1	0	1	4	0	4
		% within Detection	100.0%	0.0%	100.0%	100.0%	0.0%	100.0%	100.0%	0.0%	100.0%
	Sequence Flow	Count	0	3	3	0	0	0	0	3	3
		% within Detection	0.0%	100.0%	100.0%	0.0%	0.0%	0.0%	0.0%	100.0%	100.0%
	Sub-Process	Count	1	3	4	0	0	0	1	3	4
		% within Detection	25.0%	75.0%	100.0%	0.0%	0.0%	0.0%	25.0%	75.0%	100.0%
Total		Count	40	48	88	28	2	30	68	50	118
		% within Detection	45.5%	54.5%	100.0%	93.3%	6.7%	100.0%	57.6%	42.4%	100.0%

Fig. 2.11: Defects detection analysis on Process Modeler for Microsoft Visio.

			Well-formedness Rules			Best Practice Rules			All Rule Types		
			Detection			Detection			Detection		
			0	1	Total	0	1	Total	0	1	Total
Scope	Activity	Count	1	0	1	4	1	5	5	1	6
		% within Detection	100.0%	0.0%	100.0%	80.0%	20.0%	100.0%	83.3%	16.7%	100.0%
	Artifacts	Count	2	1	3	0	0	0	2	1	3
		% within Detection	66.7%	33.3%	100.0%	0.0%	0.0%	0.0%	66.7%	33.3%	100.0%
	Collaboration	Count	0	1	1	0	0	0	0	1	1
		% within Detection	0.0%	100.0%	100.0%	0.0%	0.0%	0.0%	0.0%	100.0%	100.0%
	Data Flow	Count	0	2	2	0	0	0	0	2	2
		% within Detection	0.0%	100.0%	100.0%	0.0%	0.0%	0.0%	0.0%	100.0%	100.0%
	Events	Count	15	12	27	8	3	11	23	15	38
		% within Detection	55.6%	44.4%	100.0%	72.7%	27.3%	100.0%	60.5%	39.5%	100.0%
	Flow Nodes	Count	0	1	1	0	0	0	0	1	1
		% within Detection	0.0%	100.0%	100.0%	0.0%	0.0%	0.0%	0.0%	100.0%	100.0%
	Gateway	Count	12	2	14	9	0	9	21	2	23
		% within Detection	85.7%	14.3%	100.0%	100.0%	0.0%	100.0%	91.3%	8.7%	100.0%
	Message Flow	Count	0	0	0	0	0	0	0	0	0
		% within Detection	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
	Process	Count	1	1	2	1	0	1	2	1	3
		% within Detection	50.0%	50.0%	100.0%	100.0%	0.0%	100.0%	66.7%	33.3%	100.0%
	Sequence Flow	Count	1	0	1	0	0	0	1	0	1
		% within Detection	100.0%	0.0%	100.0%	0.0%	0.0%	0.0%	100.0%	0.0%	100.0%
	Sub-Process	Count	1	0	1	0	0	0	1	0	1
		% within Detection	100.0%	0.0%	100.0%	0.0%	0.0%	0.0%	100.0%	0.0%	100.0%
Total		Count	33	20	53	22	4	26	55	24	79
		% within Detection	62.3%	37.7%	100.0%	84.6%	15.4%	100.0%	69.6%	30.4%	100.0%

Fig. 2.12: Defects detection analysis on jBPMN 6.

			Well-formedness Rules			Best Practice Rules			All Rule Types		
			Detection			Detection			Detection		
			0	1	Total	0	1	Total	0	1	Total
Scope	Activity	Count	6	0	6	5	0	5	11	0	11
		% within Detection	100.0%	0.0%	100.0%	100.0%	0.0%	100.0%	100.0%	0.0%	100.0%
	Artifacts	Count	0	1	1	0	0	0	0	1	1
		% within Detection	0.0%	100.0%	100.0%	0.0%	0.0%	0.0%	0.0%	100.0%	100.0%
	Collaboration	Count	0	1	1	0	0	0	0	1	1
		% within Detection	0.0%	100.0%	100.0%	0.0%	0.0%	0.0%	0.0%	100.0%	100.0%
	Data Flow	Count	3	1	4	0	0	0	3	1	4
		% within Detection	75.0%	25.0%	100.0%	0.0%	0.0%	0.0%	75.0%	25.0%	100.0%
	Events	Count	25	14	39	11	2	13	36	16	52
		% within Detection	64.1%	35.9%	100.0%	84.6%	15.4%	100.0%	69.2%	30.8%	100.0%
	Flow Nodes	Count	0	1	1	0	0	0	0	1	1
		% within Detection	0.0%	100.0%	100.0%	0.0%	0.0%	0.0%	0.0%	100.0%	100.0%
	Gateway	Count	13	1	14	8	1	9	21	2	23
		% within Detection	92.9%	7.1%	100.0%	88.9%	11.1%	100.0%	91.3%	8.7%	100.0%
	Message Flow	Count	4	0	4	1	0	1	5	0	5
		% within Detection	100.0%	0.0%	100.0%	100.0%	0.0%	100.0%	100.0%	0.0%	100.0%
	Process	Count	1	1	2	1	0	1	2	1	3
		% within Detection	50.0%	50.0%	100.0%	100.0%	0.0%	100.0%	66.7%	33.3%	100.0%
	Sequence Flow	Count	1	2	3	0	0	0	1	2	3
		% within Detection	33.3%	66.7%	100.0%	0.0%	0.0%	0.0%	33.3%	66.7%	100.0%
	Sub-Process	Count	3	1	4	0	0	0	3	1	4
		% within Detection	75.0%	25.0%	100.0%	0.0%	0.0%	0.0%	75.0%	25.0%	100.0%
Total		Count	56	23	79	26	3	29	82	26	108
		% within Detection	70.9%	29.1%	100.0%	89.7%	10.3%	100.0%	75.9%	24.1%	100.0%

Fig. 2.13: Defects detection analysis on Modelio.

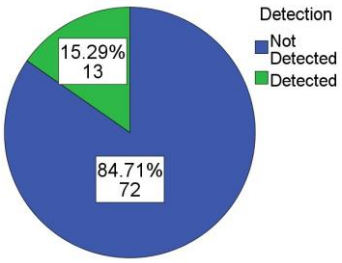
			Well-formedness Rules			Best Practice Rules			All Rule Types		
			Detection			Detection			Detection		
			0	1	Total	0	1	Total	0	1	Total
Scope	Activity	Count	3	3	6	5	0	5	8	3	11
		% within Detection	50.0%	50.0%	100.0%	100.0%	0.0%	100.0%	72.7%	27.3%	100.0%
	Artifacts	Count	2	1	3	0	0	0	2	1	3
		% within Detection	66.7%	33.3%	100.0%	0.0%	0.0%	0.0%	66.7%	33.3%	100.0%
	Collaboration	Count	1	0	1	0	0	0	1	0	1
		% within Detection	100.0%	0.0%	100.0%	0.0%	0.0%	0.0%	100.0%	0.0%	100.0%
	Data Flow	Count	0	3	3	0	0	0	0	3	3
		% within Detection	0.0%	100.0%	100.0%	0.0%	0.0%	0.0%	0.0%	100.0%	100.0%
	Events	Count	28	18	46	11	2	13	39	20	59
		% within Detection	60.9%	39.1%	100.0%	84.6%	15.4%	100.0%	66.1%	33.9%	100.0%
	Flow Nodes	Count	1	0	1	0	0	0	1	0	1
		% within Detection	100.0%	0.0%	100.0%	0.0%	0.0%	0.0%	100.0%	0.0%	100.0%
	Gateway	Count	3	9	12	9	0	9	12	9	21
		% within Detection	25.0%	75.0%	100.0%	100.0%	0.0%	100.0%	57.1%	42.9%	100.0%
	Message Flow	Count	1	3	4	1	0	1	2	3	5
		% within Detection	25.0%	75.0%	100.0%	100.0%	0.0%	100.0%	40.0%	60.0%	100.0%
	Process	Count	2	1	3	1	0	1	3	1	4
		% within Detection	66.7%	33.3%	100.0%	100.0%	0.0%	100.0%	75.0%	25.0%	100.0%
	Sequence Flow	Count	0	3	3	0	0	0	0	3	3
		% within Detection	0.0%	100.0%	100.0%	0.0%	0.0%	0.0%	0.0%	100.0%	100.0%
	Sub-Process	Count	2	2	4	0	0	0	2	2	4
		% within Detection	50.0%	50.0%	100.0%	0.0%	0.0%	0.0%	50.0%	50.0%	100.0%
Total		Count	43	43	86	27	2	29	70	45	115
		% within Detection	50.0%	50.0%	100.0%	93.1%	6.9%	100.0%	60.9%	39.1%	100.0%

Fig. 2.14: Defects detection analysis on Yaoqiang Shi.

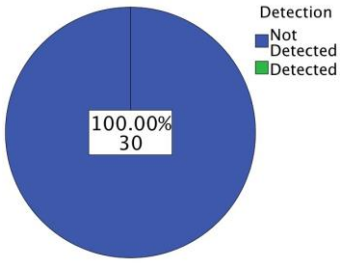
2.3.2 Model smells analysis per category and tool

The next sequence of figures from 2.15 to 2.25 represent the model smells analysis divided in each category of rules, *well-formedness* and *best practice* rules, for each tool.

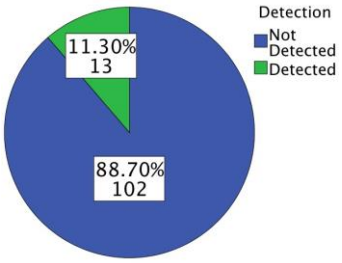
Each one of these figures is composed by three graphs : well-formedness rules detection, best practice rules detection and all rules detection, and represents the final results of model smells detection for a specific tool. The purpose of these figures is to understand in a easy way what was the final results and detection capabilities for each tool considered for analysis.



Well-formedness rules detection

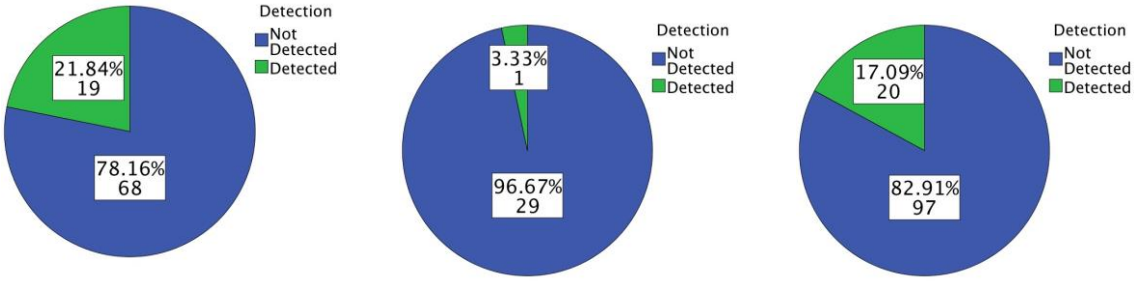


Best practice rules detection



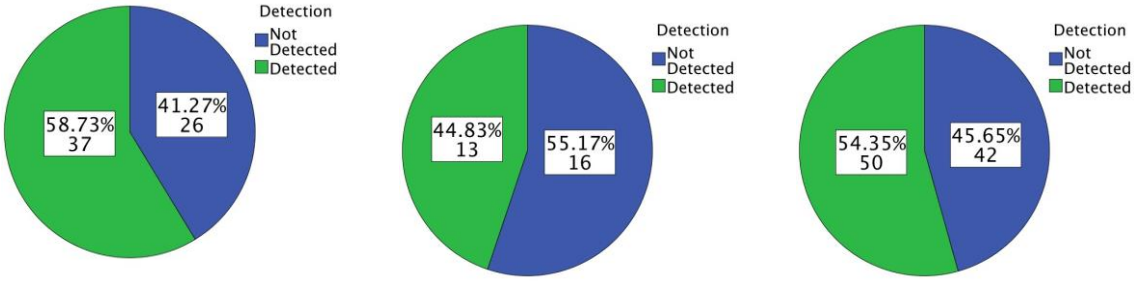
All rules detection

Fig. 2.15: Model Smells Analysis for Aris Express.



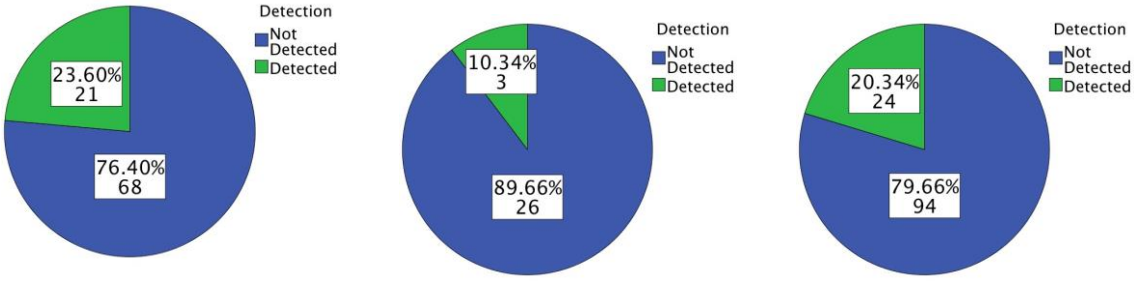
Well-formedness rules detection Best practice rules detection All rules detection

Fig. 2.16: Model Smells Analysis for BizAgi.



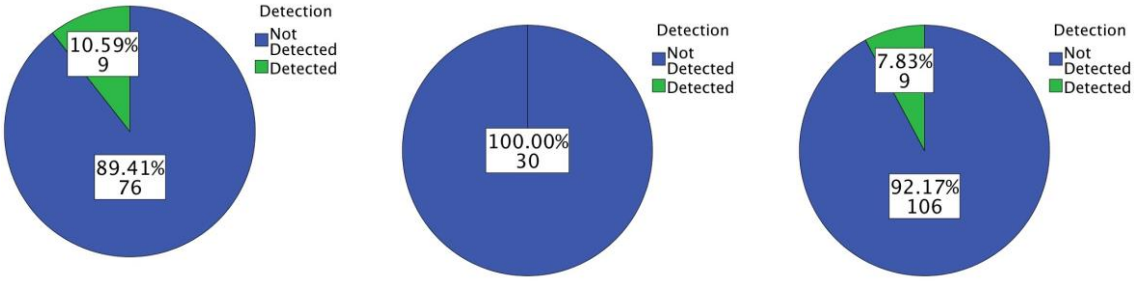
Well-formedness rules detection Best practice rules detection All rules detection

Fig. 2.17: Model Smells Analysis for Bonita Soft.



Well-formedness rules detection Best practice rules detection All rules detection

Fig. 2.18: Model Smells Analysis for Eclipse Modeler.

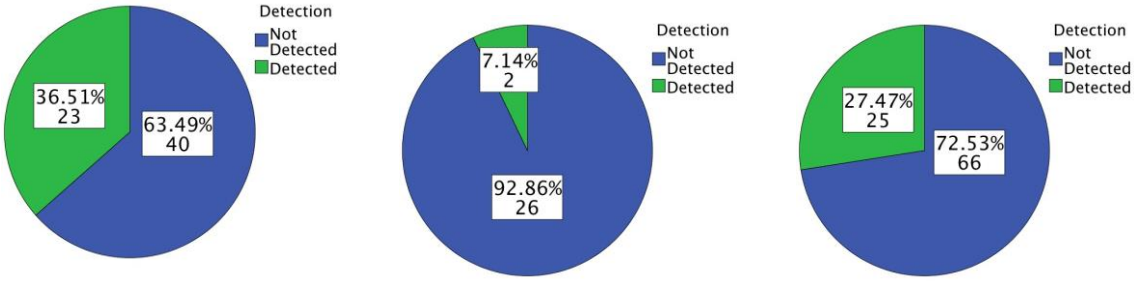


Well-formedness rules detection

Best practice rules detection

All rules detection

Fig. 2.19: Model Smells Analysis for Enterprise Architect.

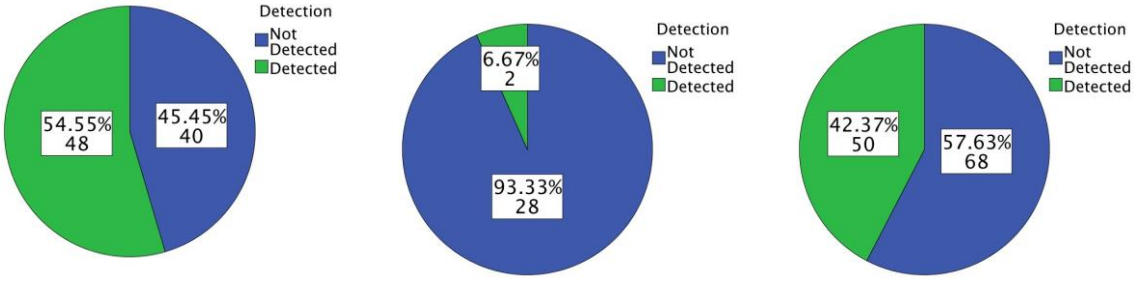


Well-formedness rules detection

Best practice rules detection

All rules detection

Fig. 2.20: Model Smells Analysis for Intalio.

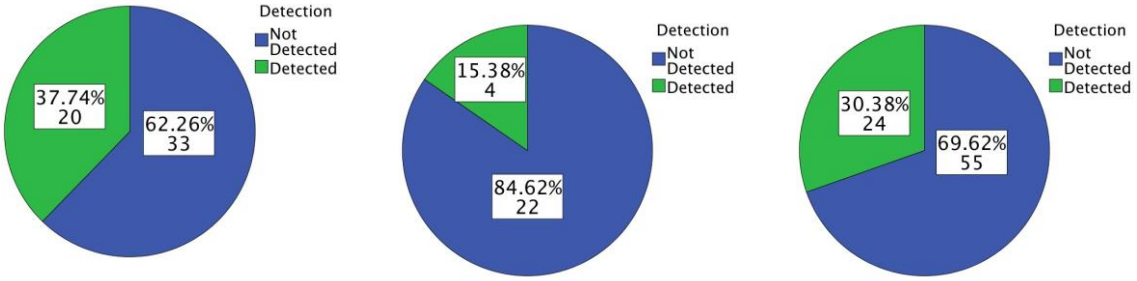


Well-formedness rules detection

Best practice rules detection

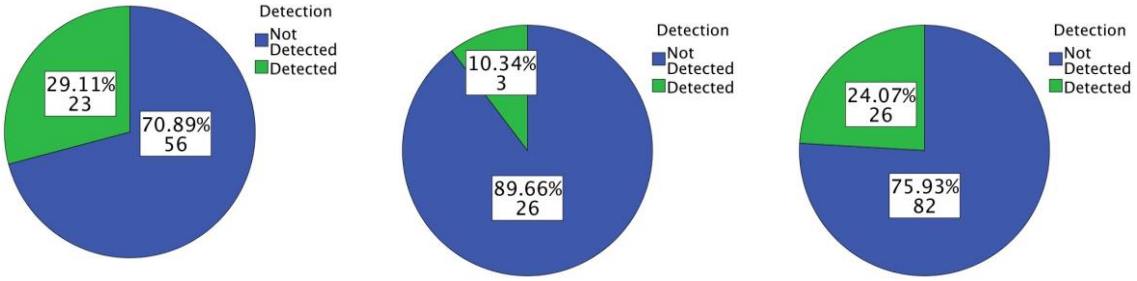
All rules detection

Fig. 2.21: Model Smells Analysis for Itp.



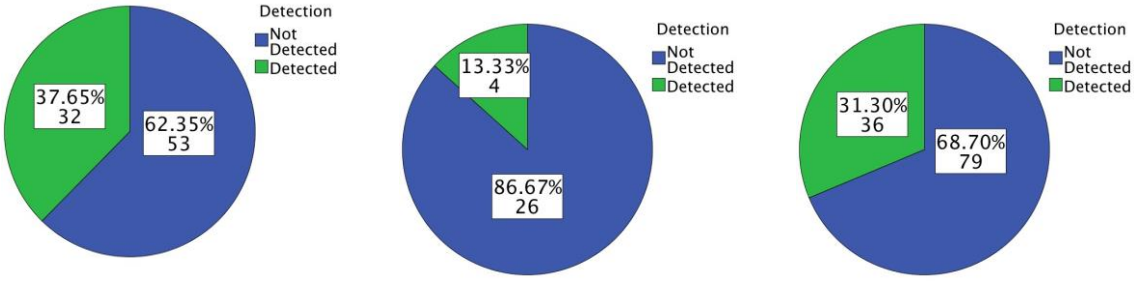
Well-formedness rules detection | Best practice rules detection | All rules detection

Fig. 2.22: Model Smells Analysis for Jbpm6.



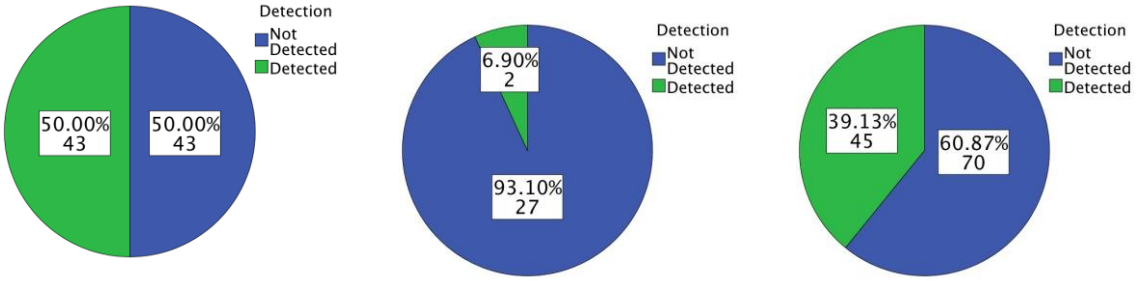
Well-formedness rules detection | Best practice rules detection | All rules detection

Fig. 2.23: Model Smells Analysis for Modelio.



Well-formedness rules detection | Best practice rules detection | All rules detection

Fig. 2.24: Model Smells Analysis for Tibco.



Well-formedness rules detection

Best practice rules detection

All rules detection

Fig. 2.25: Model Smells Analysis for Yaoqiang Shi.

2.3.3 Model smells analysis per scope

The next sequence of figures from 2.26 to 2.36 represent the model smells detection for each scope considered. The absolute values represent the total of model smells detected or not detected. The relative values (percentages) represent the total of model smells detected or not detected against the total of model smells evaluated in all tools for the specific scope.

Based on figure 2.3, it is important to remind that the scope Events represents sixty (60) and Gateway that represents twenty-two (22) of the total of model smells - one hundred and twenty-two (122).

The figure 2.26 represents the model smells detection for scope Activity. Based on this figure we can conclude that the tools with smallest number, zero (0), of detected model smells were Aris Express, Enterprise Architect and Modelio. On the other side, the tool with largest number of detected model smells , seven (7), was Bonita Soft.

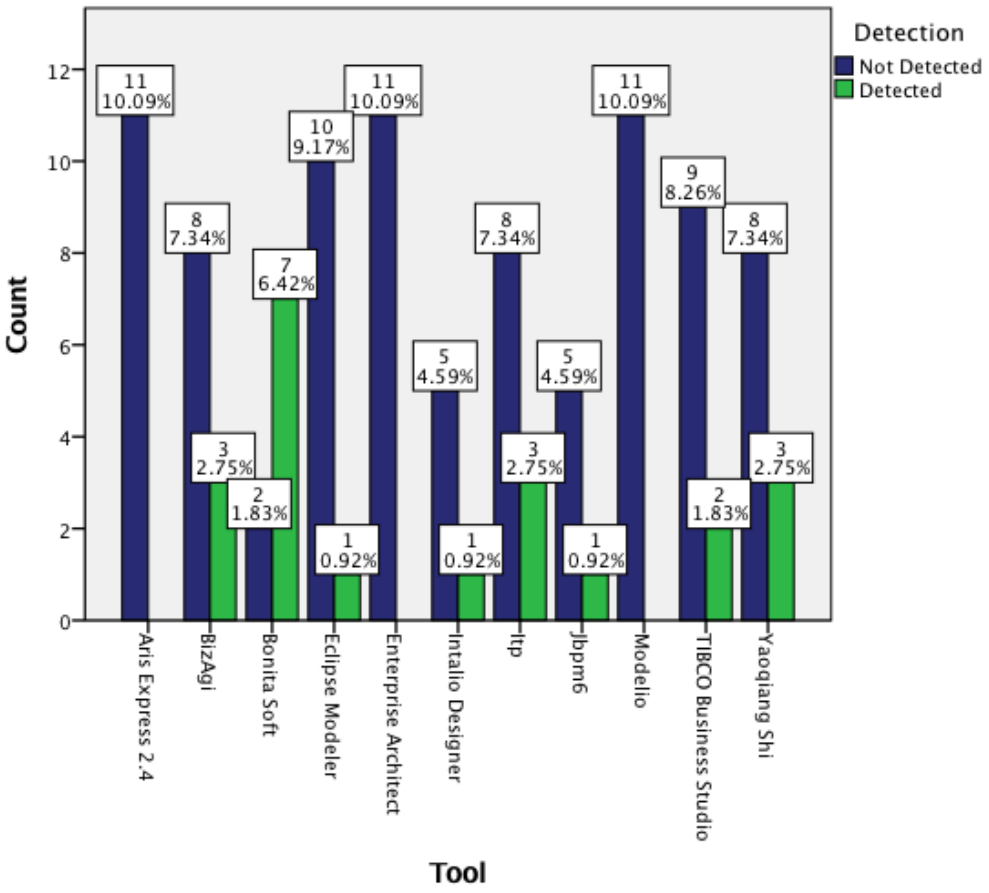


Fig. 2.26: Number of defects not detected by Scope - Activity

The figure 2.27 represents the model smells detection for scope Artifacts. Based on this figure we can conclude that the tools with smallest number, one (1), of detected model smells were Jbpm6, Modelio and Yaoqiang Shi. On the other side, the tool with largest number of detected model smells , three (3), was Itp.

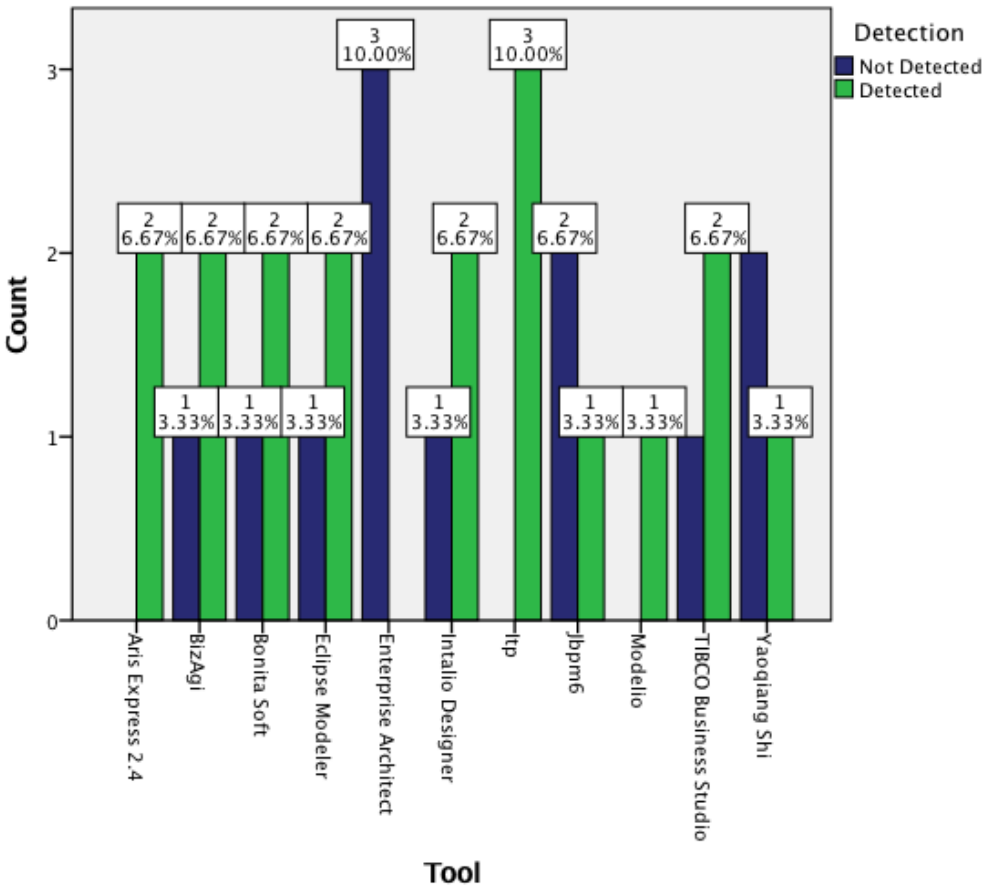


Fig. 2.27: Number of defects not detected by Scope - Artifacts

The figure 2.28 represents the model smells detection for scope Collaboration. Based on this figure we can conclude that there is several tools with smallest number, zero (0), of detected model smells. On the other side, there is several tools with largest number, one (1), of detected model smells.

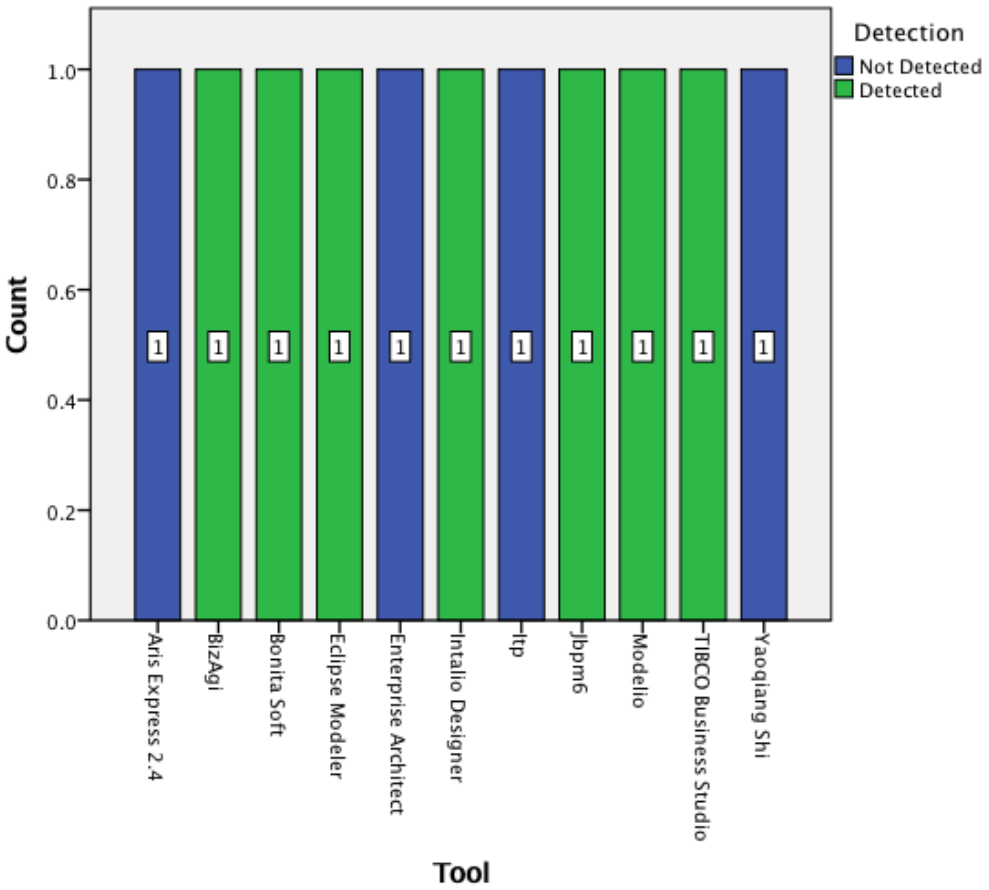


Fig. 2.28: Number of defects not detected by Scope - Collaboration

The figure 2.29 represents the model smells detection for scope Data Flow. Based on this figure we can conclude that there is several tools with smallest number, zero (0), of detected model smells. On the other side, the tool with largest number of detected model smells , three (3), was Yaoqiang Shi.

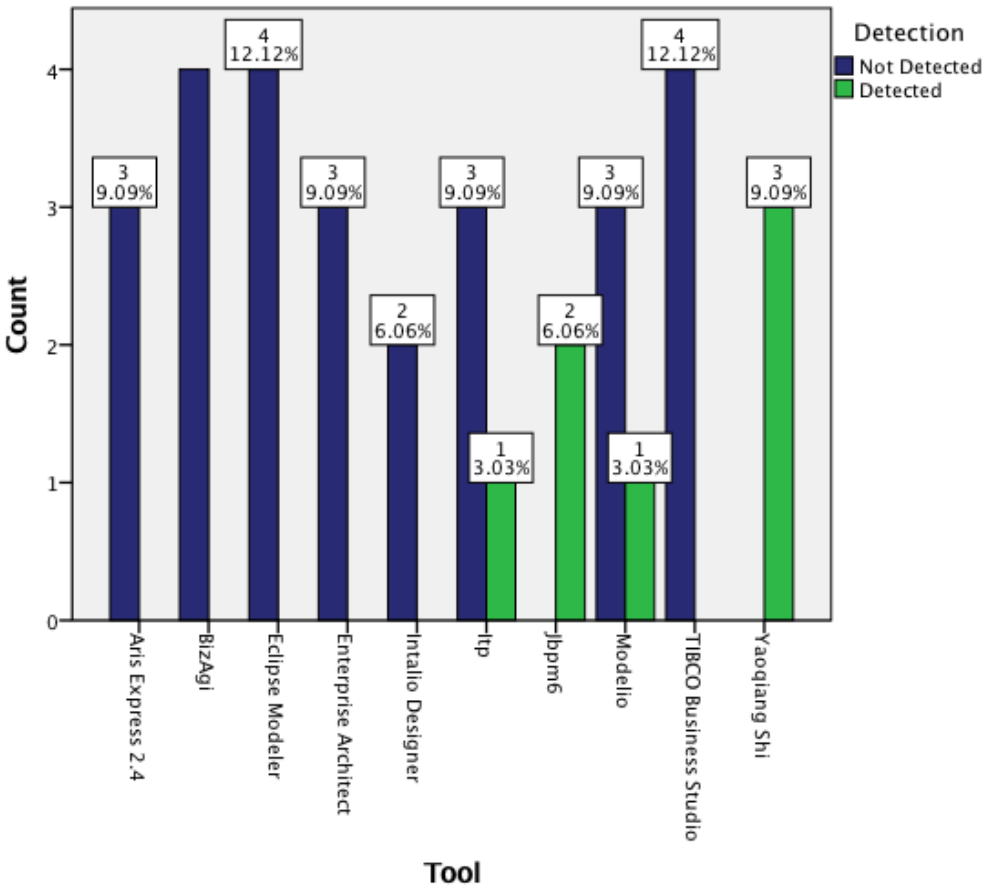


Fig. 2.29: Number of defects not detected by Scope - Data Flow

The figure 2.30 represents the model smells detection for scope Events. Based on this figure we can conclude that the tool with smallest number, six (6), of detected model smells was Enterprise Architect. On the other side, the tool with largest number, twenty-seven (27), of detected model smells, was Itp.

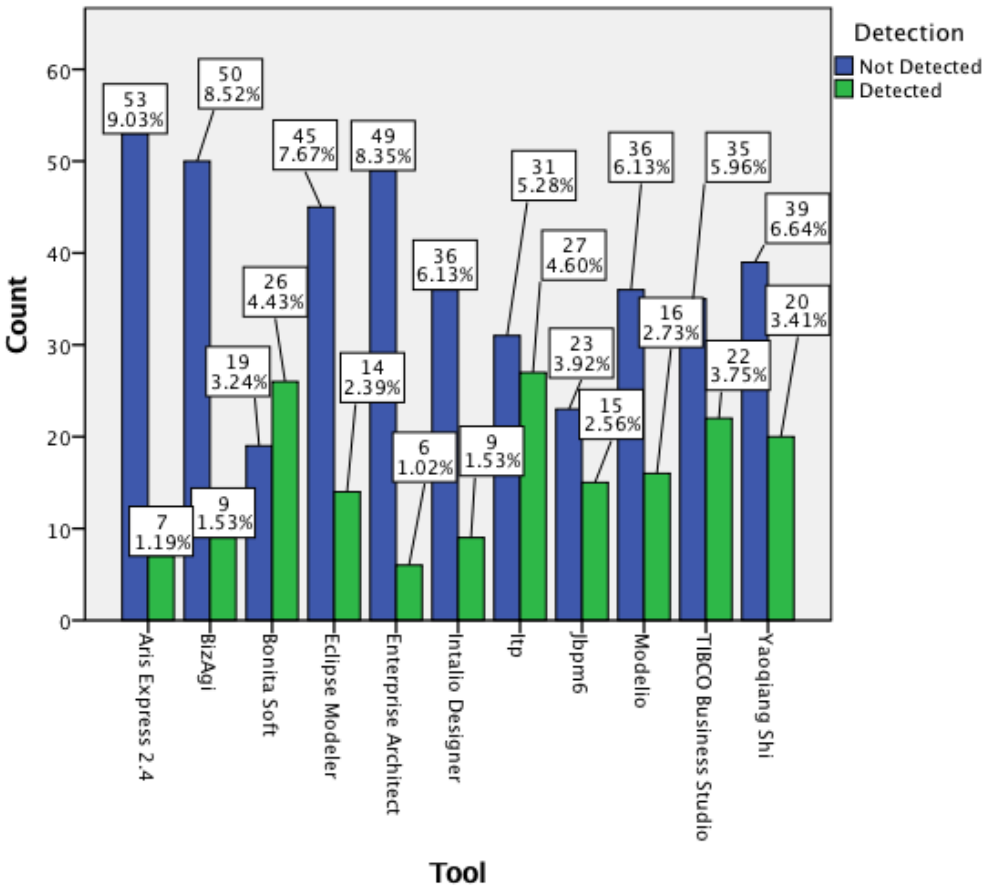


Fig. 2.30: Number of defects not detected by Scope - Events

The figure 2.31 represents the model smells detection for scope Flow Nodes. Based on this figure we can conclude that there is several tools with smallest number, zero (0), of detected model smells. On the other side, there is several tools with largest number, one (1), of detected model smells.

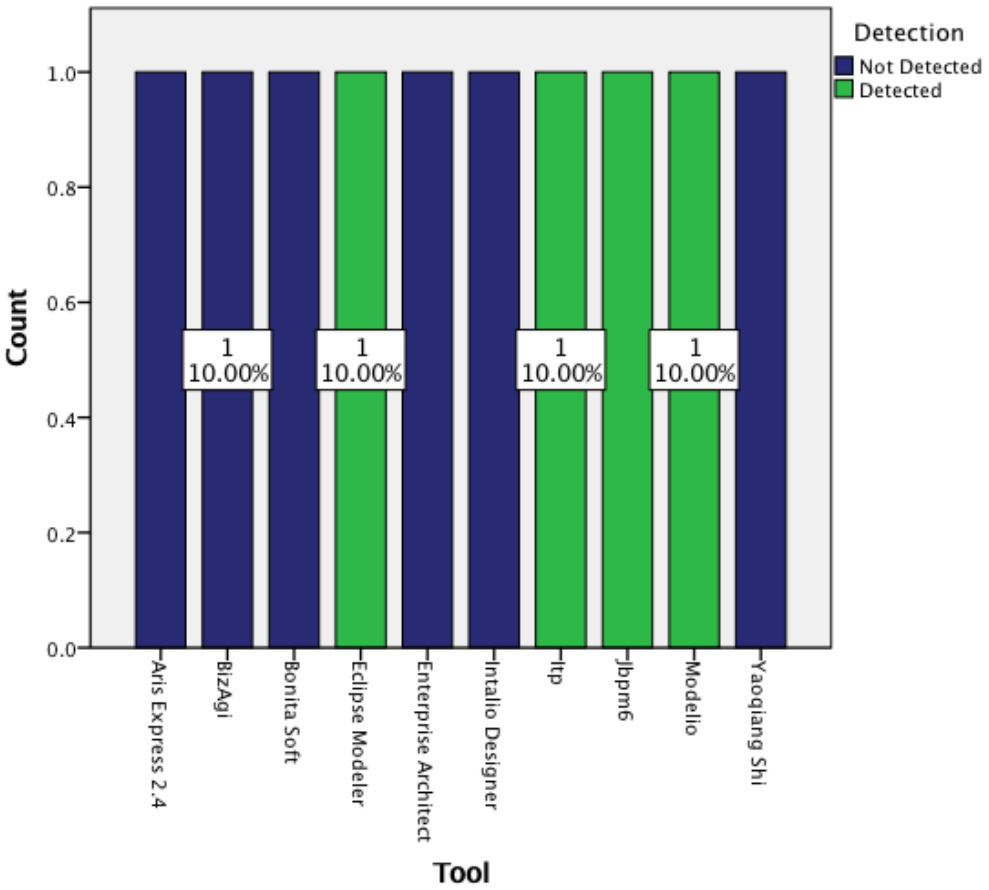


Fig. 2.31: Number of defects not detected by Scope - Flow Nodes

The figure 2.32 represents the model smells detection for scope Gateway. Based on this figure we can conclude that the tools with smallest number, zero (0), of detected model smells were Aris Express, BizAgi and Enterprise Architect. On the other side, the tools with largest number, nine (9), of detected model smells, were Itp and Yaoqiang Shi.

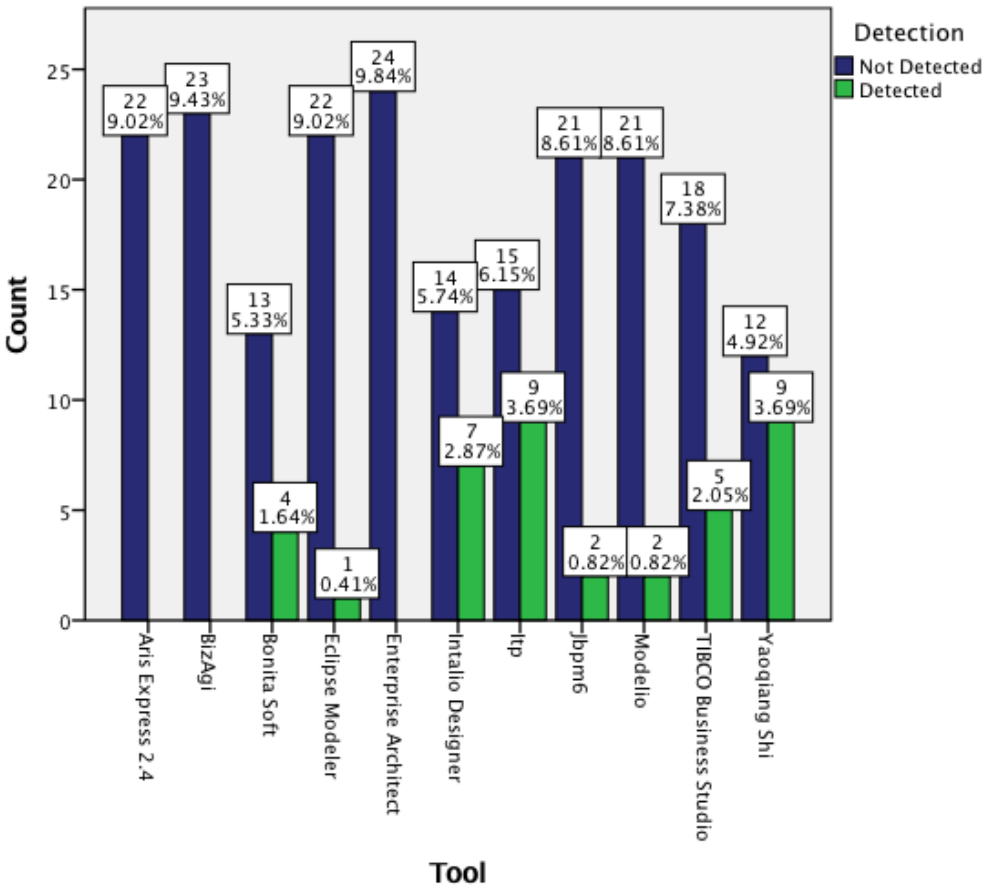


Fig. 2.32: Number of defects not detected by Scope - Gateway

The figure 2.33 represents the model smells detection for scope Message Flow. Based on this figure we can conclude that there is several tools with smallest number, zero (0), of detected model smells. On the other side, the tool with largest number, four (4), of detected model smells, was Bonita Soft.

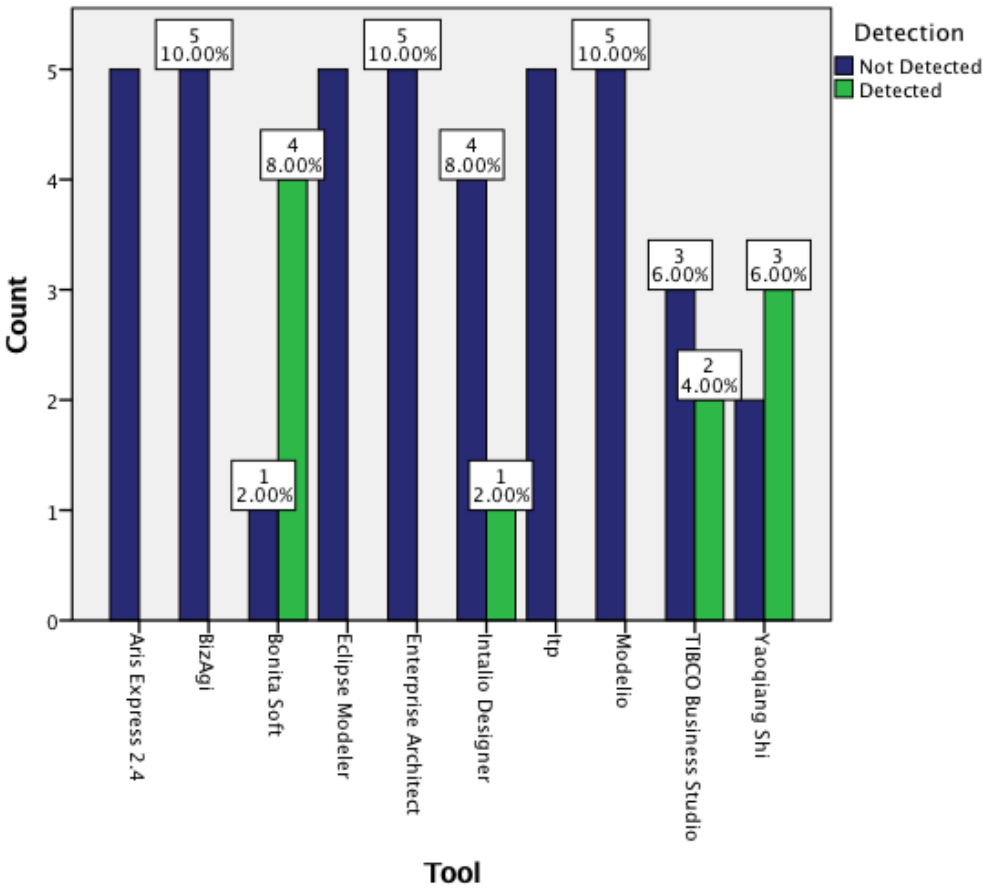


Fig. 2.33: Number of defects not detected by Scope - Message Flow

The figure 2.34 represents the model smells detection for scope Process. Based on this figure we can conclude that the tools with smallest number, zero (0), of detected model smells were Enterprise Architect and Itp. On the other side, the tool with largest number, two (2), of detected model smells, was Aris Express.

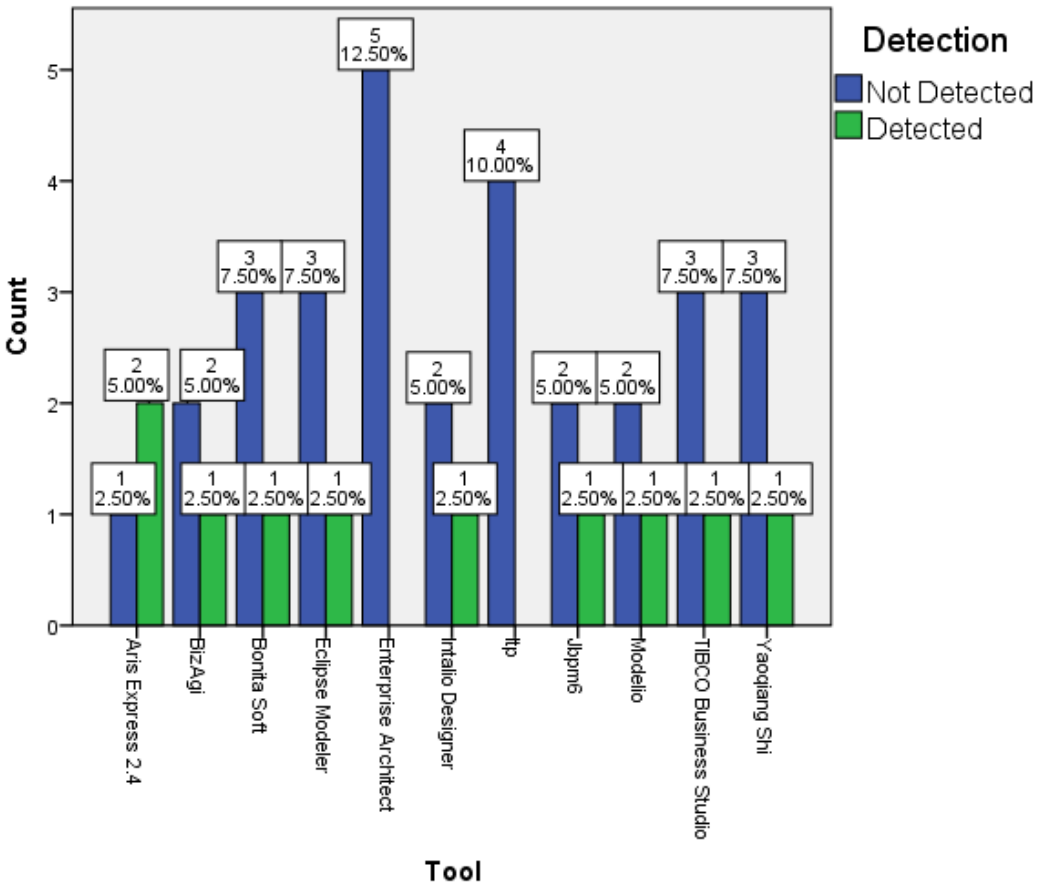


Fig. 2.34: Number of defects not detected by Scope - Process

The figure 2.35 represents the model smells detection for scope Sequence Flow. Based on this figure we can conclude that the tools with smallest number, zero (0), of detected model smells were Aris Express and jBPM6. On the other side, the tools with largest number, three (3), of detected model smells, were Intalio Designer, Itp and Yaoqiang Shi.

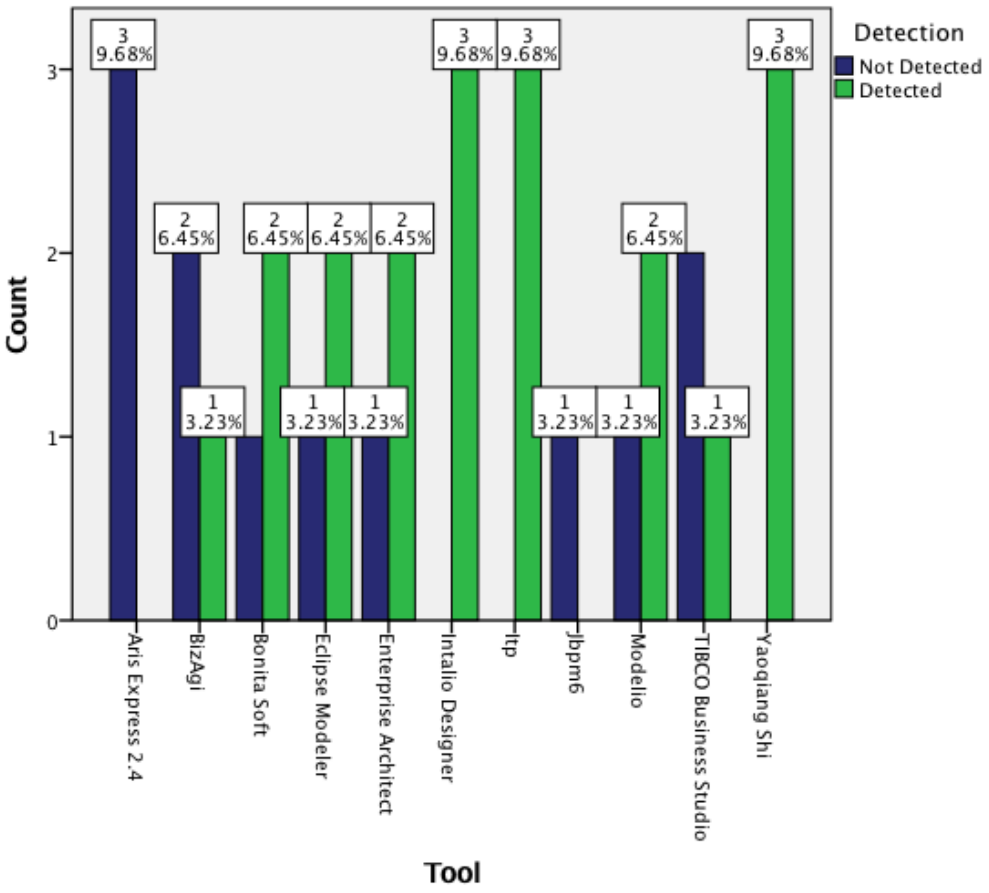


Fig. 2.35: Number of defects not detected by Scope - Sequence Flow

The figure 2.36 represents the model smells detection for scope Sub-Process. Based on this figure we can conclude that the tools with smallest number, zero (0), of detected model smells were Intalio Designer, jBPM6 and Tibco Business Studio. On the other side, the tools with largest number, three (3), of detected model smells, were Bizagi, Bonita Soft and Itp.

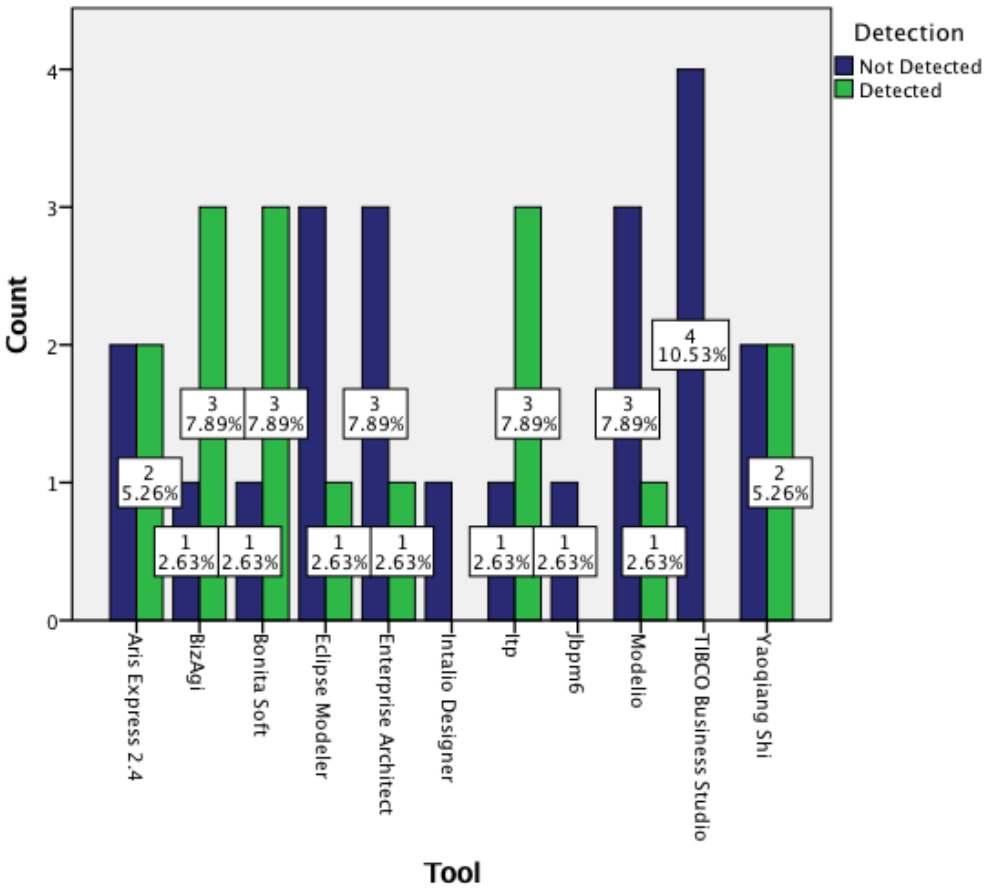


Fig. 2.36: Number of defects not detected by Scope - Sub-Process

The figure 2.37 shows the total number of well-formedness smells that were not detected for each scope of BPMN2 elements. We can see that the two scopes with most of defects not detected are Events and Gateways.

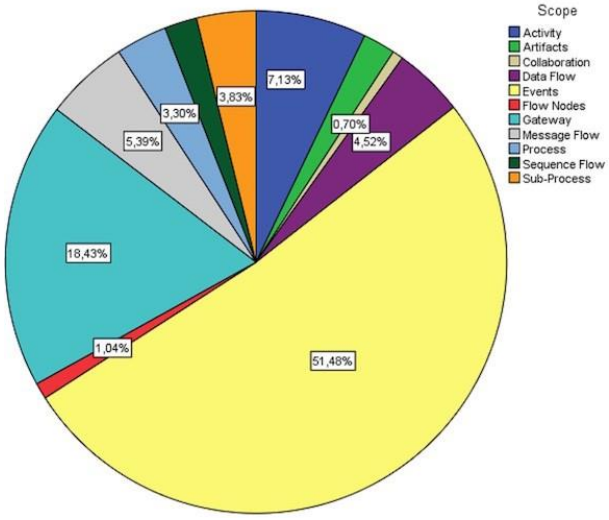


Fig. 2.37: Number of defects not detected by Category - Well-formedness Rules

The figure 2.38 shows the total number of best practices smells that were not detected for each scope of BPMN2 elements. We can see that the two scopes with most of defects not detected are Events and Gateways.

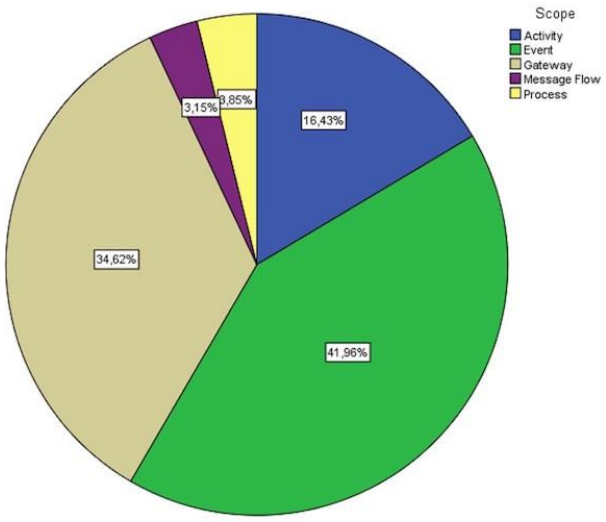


Fig. 2.38: Number of defects not detected by Category - Best Practices

The figures 2.39 and 2.40 show the total of rules successfully analysed for each tool, and the number of model smells detected and not detected. Additionally, in figure 2.39, at right we have the percentage of success and failure for each tool. The percentages were calculated based on the number of rules detected/not detected by the tool against the number of rules tested in the tool.

		Detection				
		Not Detected	Detected	Total	Success (%)	Failure (%)
Tool	Aris Express 2.4	102	13	115	11%	89%
	BizAgi	97	20	117	17%	83%
	Bonita Soft	42	50	92	54%	46%
	Eclipse Modeler	94	24	118	20%	80%
	Enterprise Architect	106	9	115	8%	92%
	Intalio Designer	66	25	91	27%	73%
	Itp	68	50	118	42%	58%
	Jbpm6	55	24	79	30%	70%
	Modelio	82	26	108	24%	76%
	TIBCO Business Studio	79	36	115	31%	69%
	Yaoqiang Shi	70	45	115	39%	61%
Total		861	322	1183		

Fig. 2.39: Model smells detection by Tool.

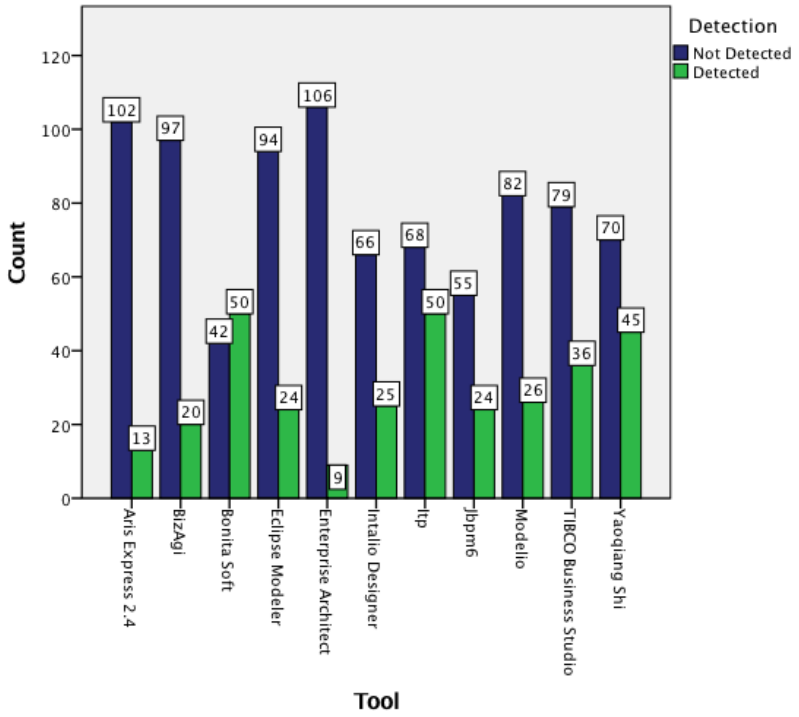


Fig. 2.40: Model smells detection by Tool.

2.4 Characterization of the state of art

In figure 2.37 we can conclude that inside the well-formedness rules category, the larger number of defects not detected by the tools belong to the scope Events. The same is verified for the best practices category, as we can see in figure 2.38. On the other side, based on figure 2.3, we can see that the scope Events contains sixty (60) rules and represents almost half of the total of model smells considered to this analysis (122). Based on this fact, it is easier to this scope has the largest number of defects not detected.

On the other side, for the category of well-formedness rules, and based on figure 2.37, we conclude that the scope of *Collaboration* had the smallest number of defects not detected, whereas this number represents zero and seventy hundredths (0.70) percent of the total number of defects analysed in all tools. In terms of best practice rules category, based on figure 2.38, we conclude that the scope with smallest number of defects not detected was *Message Flow*, whereas it represents three and fifteen hundredths (3.15) percent of the total number of defects analysed in all tools.

In figure 2.39 we can see that the tool with the least number of defects not detected is Bonita Soft for both categories of defects, and consequently with the highest percentage of success in terms of defects detection (54%). On the other hand, we have Itp tool with

the same number of defects detected (50), but since this tool had more rules analysed in total (118) in comparision with Bonita Soft (92), consequentely Itp tool had a smaller percentage of success in model smells detection (42%).

In terms of the tool with the least validation capabilities, based in figure 2.39, we have the Enterprise Architect tool with the highest number of not detected defects (9). If we take a look in detail for the two model smells categories, based in figure 2.10, we conclude that Enterprise Architect was the tool with smallest number of defects detected for well-formedness and best practice rules, whereas it had nine (9) and zero (0) defects detected respectively.

In conclusion, we performed an analysis of defects for a specific set of tools. These defects are represented in two main categories: well-formedness rules and best practices rules. In general, the tool that had highest number of detected defects was Bonita Soft. Then, looking to the data inside each category, for the well-formedness category we have Enterprise Architect as the tool with the smallest number of detected defects, and Itp Process Modeler and Bonita Soft as the tools with highest number of detected defects. For the best practices category, we have Aris Express and Enterprise Architect as the tools with smallest number of detected defects and Bonita Soft as the tool with the highest number of detected defects.

[This page was intentionally left blank]

3. BPMN MODEL CHECKING TOOL PROTOTYPE

3.1 The OMG Model-Driven Engineering framework

The model-driven engineering is a methodology in software development area which focus is create and exploit domain models, which are conceptual models to all aspects related to a specific problem. One of the MDE initiatives is model-driven architecture (MDA) from OMG - figure 3.2.

In the figure 3.1, there is a representation of all the topics related with Model-Driven Engineering (MDE) and even from which topics is derived the OMG's model driven architecture.

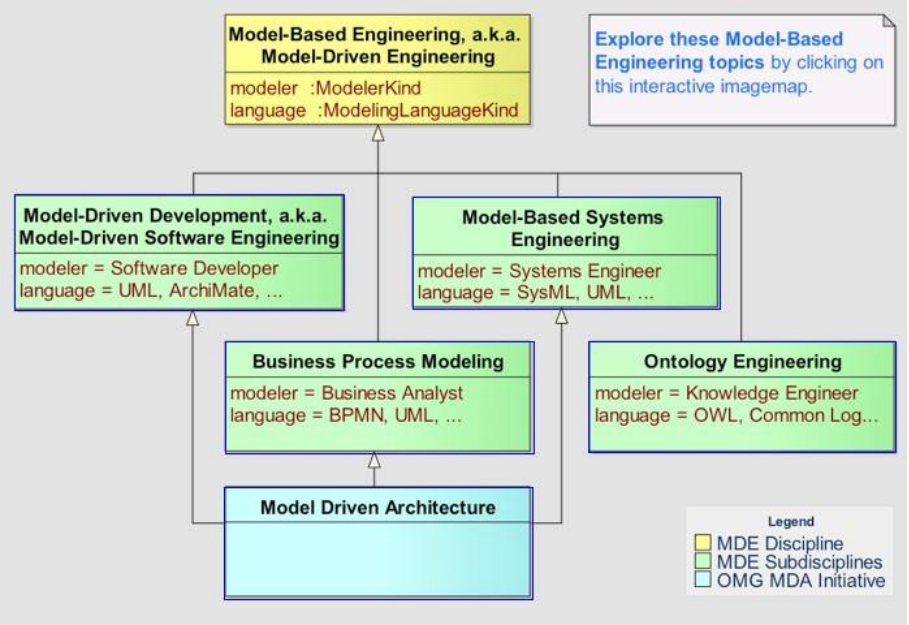


Fig. 3.1: Model-Based Engineering Concepts [1].

[1](source: Model-Based Engineering Forum)

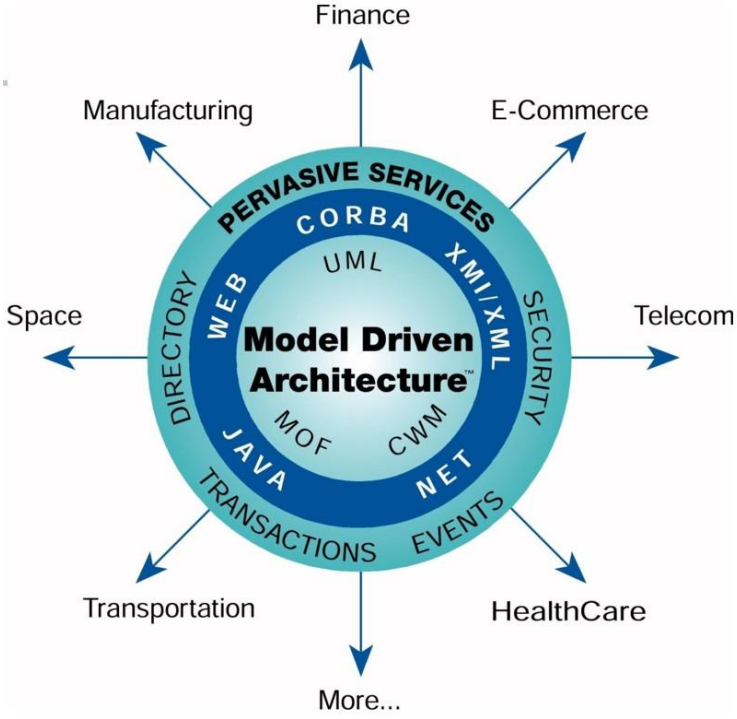


Fig. 3.2: Model-Driven Architecture [2].

3.2 The BPMN metamodel

The BPMN standard specification [10] is an official document that specificies each BPMN's element and their connections and meanings. Besides, it is considered a complex technical document to business modelers. Therefore, there was the need to create a BPMN metamodel (considered inside M2 level, according to the MDA paradigm), easier to understand and definitions and rules informally presented in plain English. It describes the abstract syntax of BPMN by means of meta-classes, meta-associations and cardinality constraints.

Based on figure 3.3, we can see the Model Driven Architecture (MDA), defined by OMG, and based on a four-layer metamodeling architecture. Each layer represents a level of abstraction of information. The BPMN specification is considered in level M1, while the BPMN metamodeling is considered in level M2. This means the BPMN metamodeling has a different level of abstraction, not referring some secondary details, and consequently being easier to understand. As mentioned in [11], there are known weaknesses in the BPMN standard that hinders the design of process models with good quality us- ing currently available tools, since BPMN specification is expressed in natural language,

[2](source: OMG Official Website)

and is vulnerable to the user's interpretation, who is reading the specification in order to express the rules and elements defined to the modeling tool(s).

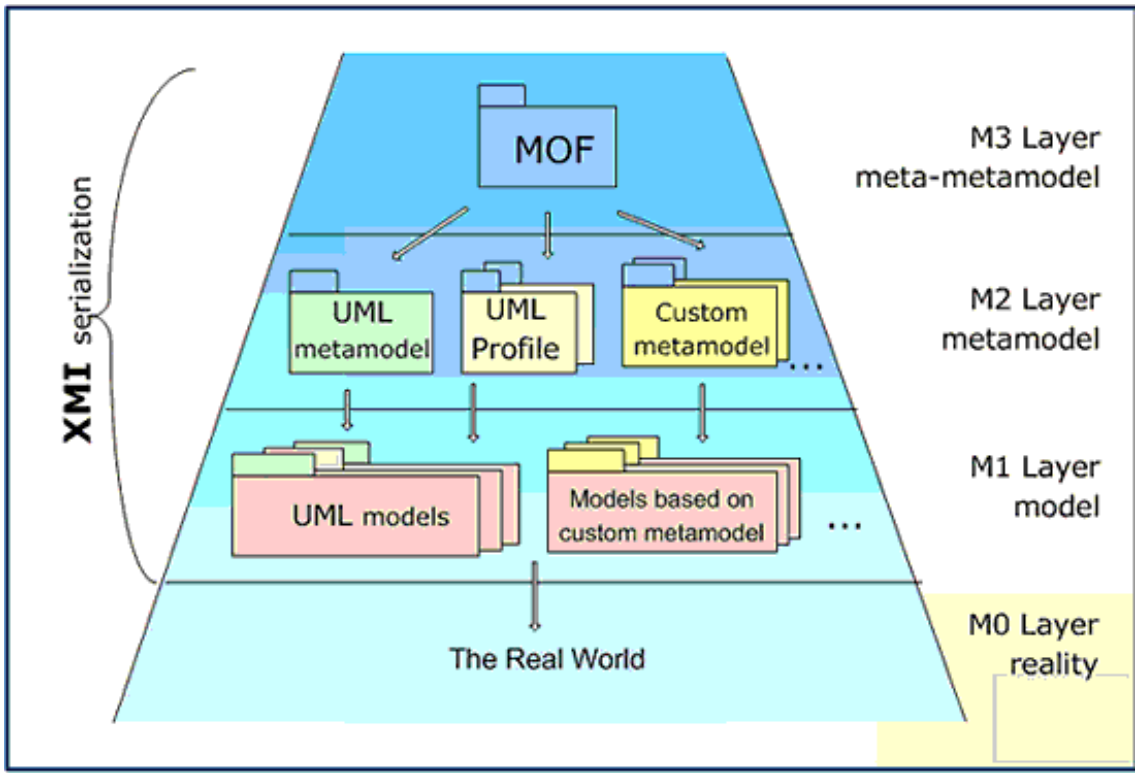


Fig. 3.3: MDA four-layer MOF-based metadata architecture [3].

3.3 Well-formedness rules and best practices

As examples of these rules, we have here two, expressed as OCL rules [11], from the set of model-checking rules expressed in BPMN2 specification and mapped to the BPMN2 metamodel.

3.3.1 Outgoing Sequence Flow not allowed in an End Event.

The end event indicates where a process will end. In terms of sequence flows, the end event ends the flow of the process, and thus, will not have any outgoing sequence flows.

In order to validate this rule, the metamodel has an invariant declared that validates if the number of outgoing sequence flows is zero for all the end events contained in the container/process.

[3](source: The Tao of Modeling Spaces)

```
inv endEventsHaveNoOutgoingSequenceFlow:
self.totalContainerEndEvents()
->forAll(numberOutputSequenceFlows() = 0)
```

3.3.2 Catch Error Event must trigger an exception flow

Error intermediate events cannot be used within normal sequence flows. This catching event should trigger an exception flow. An error is caught by an error intermediate event attached to the sub-process border. The intermediate event should trigger an exception flow.

```
inv catchErrorEventTriggerExceptionFlow:
self.isErrorEvent() implies
(self.oclAsType(BoundaryEvent).attachedToRef
.isDefined() and
self.outgoing a.targetRef->notEmpty())
```

3.4 BPMN models portability - The XPDL Standard

The XML Process Definition Language (XPDL) is a format specified by the Workflow Management Coalition (WfMC) to specify business process models [21]. XPDL is represented by a XML schema that expresses the constraints applied to the content represented in this format. XPDL 2.2 has been created to express all aspects of a BPMN diagram ensuring the support for BPMN2. Based on authors of [21], XPDL supports every aspect of BPMN, including graphical aspects and run time properties. So, currently any business process modeling tool with support to XPDL can export the process definition to XPDL files, even it can be import to another tool with XPDL support without lose properties of the diagram, preserving the diagram as it is originally.

One of the principles of WfMC is to keep XPDL up to date with BPMN, and the lastest version (2.2) of XPDL was specified in order to specify BPMN 2.0 business process models. Since we designed a program to read the XPDL content of the files exported from the BPMN2 modeling tool, and identify the different BPMN elements and properties specified, we have to worry about mapping between XPDL 2.2 and BPMN 2.0. In initial versions of XPDL, there were problems with mapping between BPMN and XPDL elements, as claimed by the authors of [22]. In the latest versions of XPDL (2.2) and BPMN

(2.0.2) we did not find any sources claiming this mapping problem. On the other side, the authors of XPDL (WfMC) claim that nowadays there is full mapping between XPDL and BPMN, as we can read from the XPDL Official Website[4]: *"Xpdl provides a file format that supports every aspect of the bpmn process definition notation including graphical descriptions of the diagram, as well as executable properties used at run time"*.

As we can see in table 3.1, XPDL is one of the most used format in the set of tools considered. Secondly, XPDL is considered by WMC, as the leading process definition language used to store and exchange process models. Therefore, XPDL was the format used to export the models designed in the BPMN modeling tools considered. The objective of export the model is to have the content of the model available in a format that we can import to our parser and manage this information in order to execute our validation process. The parser will be able to read the content of the exported files, identify each one of the BPMN elements that is contained in the model, instantiate the BPMN Metamodel in the USE environment. Thereby, the validator controller will load the BPMN2 metamodel using commands in USE environment, and validate the content of the model. The validation of the model is done by OCL invariants, that are basically logical expressions that express the rules contained in the BPMN2 specification or best practice rules formulated by BPMN modeling experts in books such as [17], [16] and [15]. The complete list of invariants defined in BPMN2 metamodel can be consulted in appendix E.

Table 3.1: BPMN modeling tools exporting formats.

Tool Name	**Export Formats Available**
Aris Express	PDF; Image files; EMF; ADF
Bizagi Process Modeler	Microsoft Visio; Image files; XPDL 2.2
Bonita BPM	BPM2 Archive, BOS Archive, Executable Business Archive, Image file
Eclipse BPMN2 modeler	Image file
Enterprise Architect	BPMN 2.0 XML, XPDL 2.2
Intalio BPMs	Image file; PDF
jBPMN	BPMN 2.0 XML, JSON
Modelio	UML/EMF 3.0.0, OMG 2.1.1, OMG 2.2 and OMG 2.3
Process Modeler for Microsoft Visio	XPDL 2.1, BPEL, XLANG/s, BPMN 2.0
Tibco Business Studio - BPM Edition	Archive File; Modelled Application Archive; Work Data Model, XPDL [5]
Yaoqiang BPMN Editor	XPDL 2.1, XML, VML, SVG, JPG, PNG and PDF format

[4]http://www.xpdl.org/

3.5 BPMN Model checking process

This section is related with all the process of usage of the metamodel-based validation facility in order to achieve the results expected.

1) Design a BPMN model in the Business Process Modelling Tool

In this step we use a BPMN Modelling Tool to design our business process model. One restriction associated with the choice of the tool is related with the functionality of export the model for a different format. Since we need to export the model to XPDL 2.2, the tool has to allow the user to export the designed model to this format. In the figure 3.4, we can see the interface of the Bizagi Tool[6]and a small example of a business process model designed.

[5]Tibco Modeling User's Guide - When user creates a project, TIBCO Business Studio creates a package and each package is represented by a XPDL file with all its content.

[6]http://www.bizagi.com/en/component/bizagicloudaccess/?task=login&tmpl=component&prd=mdl

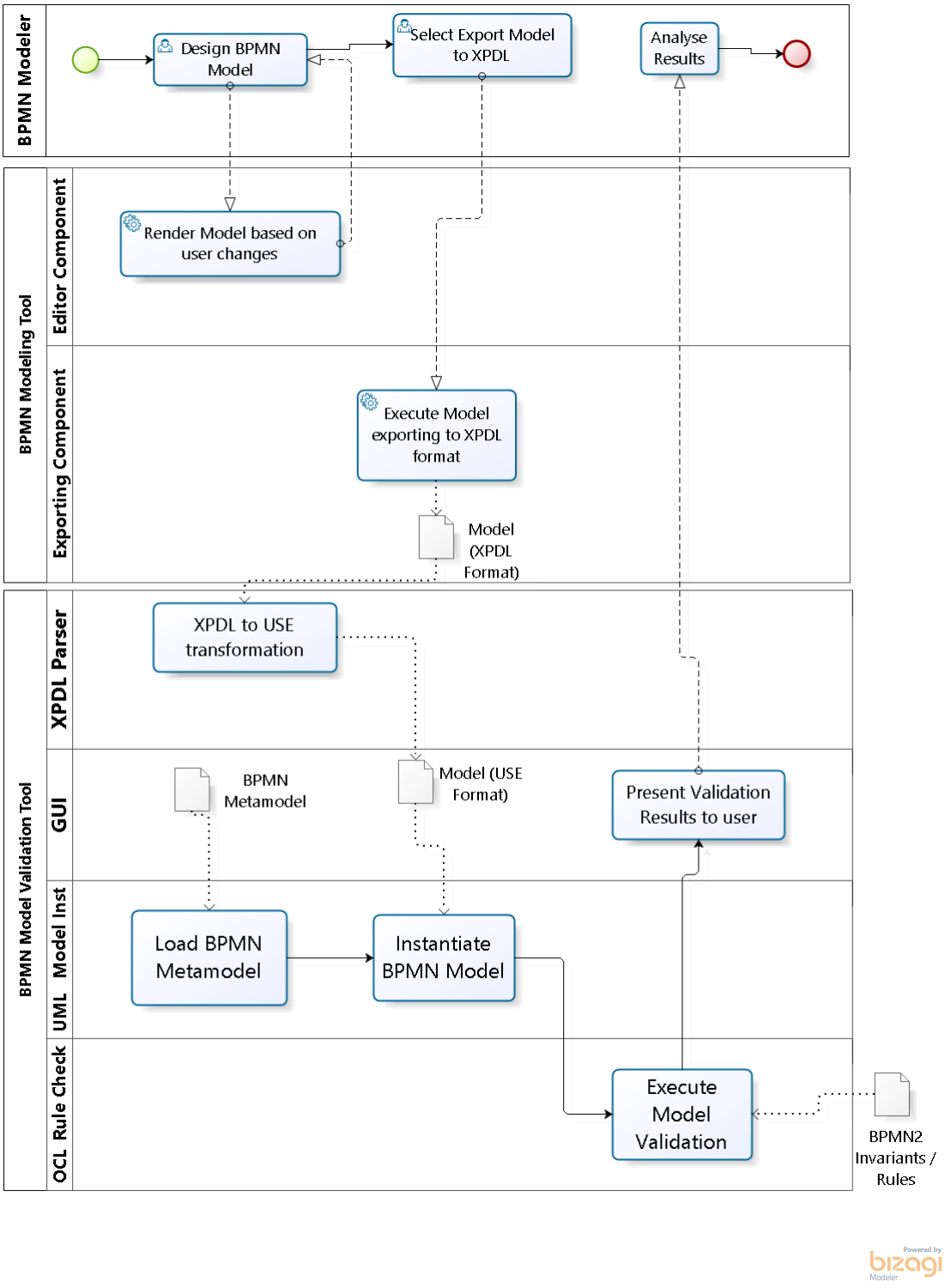


Fig. 3.4: Bizagi Tool - Design of an example model

2) Export the BPMN model

In this step we export the designed model to the XPDL 2.2 format. In the figure 3.5 we can see that Bizagi tool has a top menu where there is an option *Export* and we

can choose the format that we want, in this case XPDL. This option generates one file with extension *.xpdl* per process. Since our example model has only one main process, the tool will generate only one file.

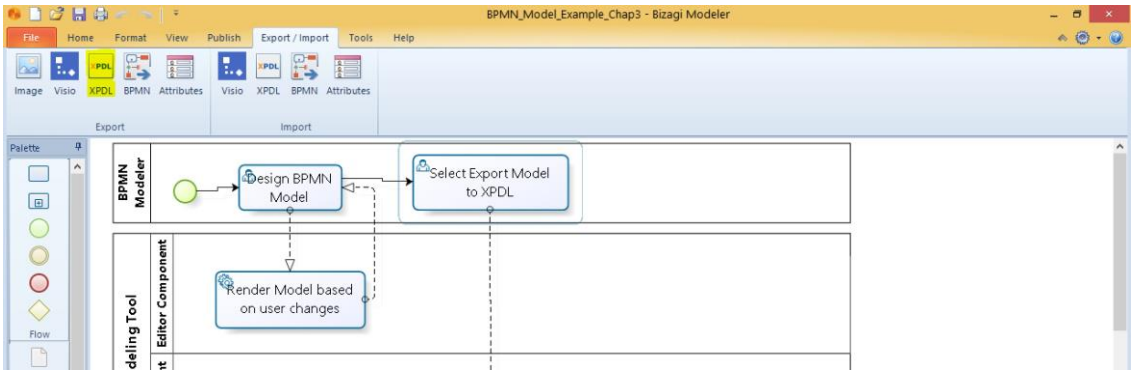


Fig. 3.5: Bizagi Tool - Export model to XPDL 2.2 format

3) Run the parser program

In this step we run the parser program in order to read the content inside the XPDL files and generate the output file with all the USE commands necessary to instantiate the business process model created in step 1 into USE environment. The parser program is composed by an executable jar file, a script responsible to execute the jar file and the BPMN metamodel file, which name is *BPMN2.0_OMG.use*. Initially, we have to ensure that the USE path defined in the script file is set for the root folder of the USE environment installation in the machine where the parser program will run, since the parser program depends of USE environment. Thereafter, we run the script file *parser.bat*, so the executable jar will execute and start the execution of the parser program.

The parser program is composed by a graphical interface, where the user can select the working folder and initiate the validation by clicking the button *Validate*. The Working folder must contain the set of XPDL files related with the BPMN model that we want validate. In figure 3.6 we can the folder structure of parser program and the files required in order to run it. In figure The discussed details are shown in next figure 3.7. In figure 3.6 we have the XPDL files in same folder as the parser, but these files can be in a different folder, and in this case it is necessary to press button *Choose working folder* and select the specific folder.

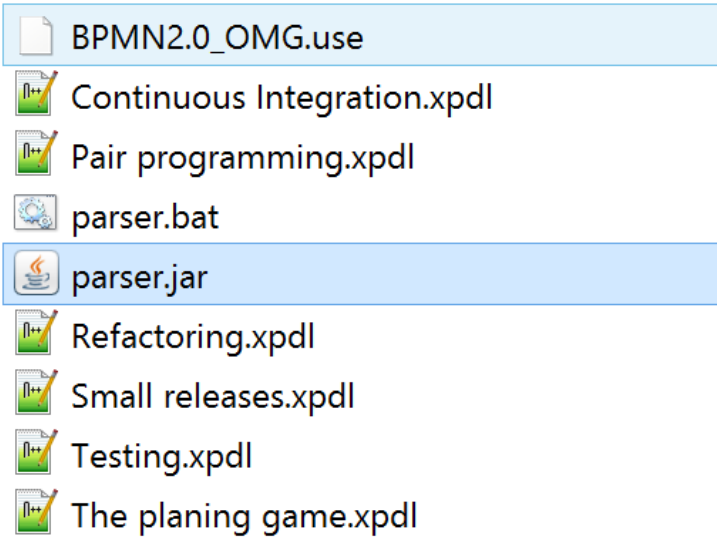


Fig. 3.6: Parser folder structure.

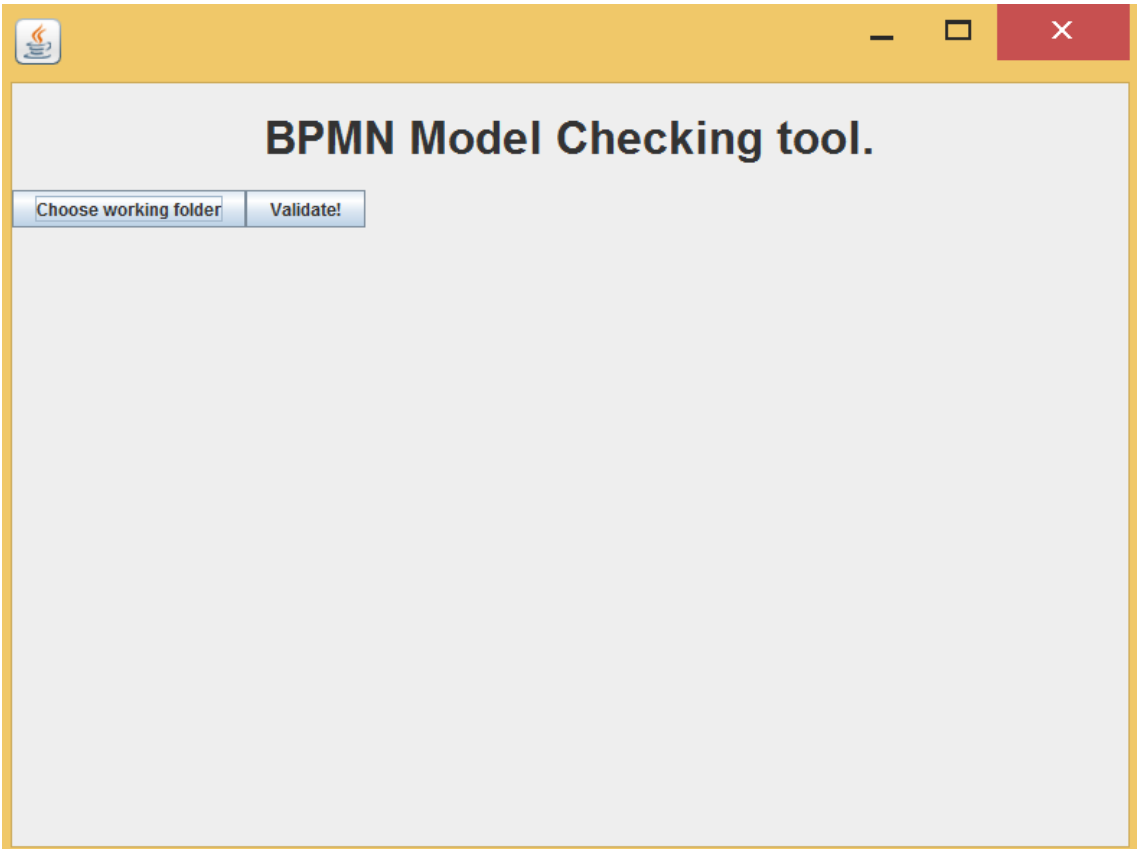


Fig. 3.7: Parser GUI - Choose Working folder.

4) Analyse validation results

After the parser finishes the BPMN model validation, there is information related

with validation results that are shown to the user. This information starts with two labels showing the total number of invariants analysed, and the total number of invariants that failed its validation. Additionally, it is presented a table with a list of all the invariants which failed their validation. Every invariant is composed by its Scope and Name. The user can any time hit the button *X* in order to close the application. On the other side, the user can run the validation several times, for the same model, or even for different model, whereas the XPDL files are changed for the ones related with the new model. The discussed details are shown in next figure 3.8. Additionally, for each of the invariants failed shown in the list, the user can double-click on any one of the list results and see description related with the invariant failed, in order to understand what the invariant is validating and how can fix it. The aspect of the message is in figure 3.9.

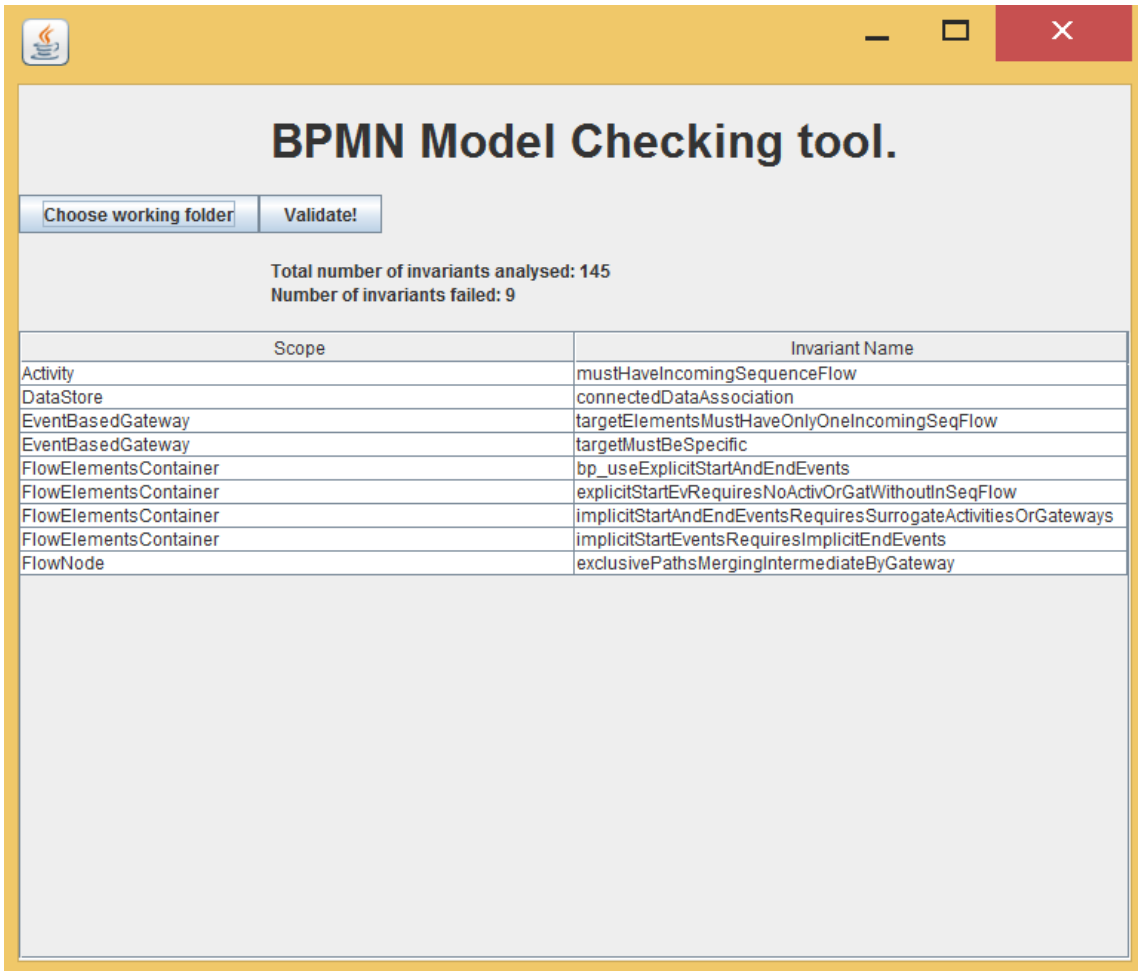


Fig. 3.8: Parser GUI - Validation Results.

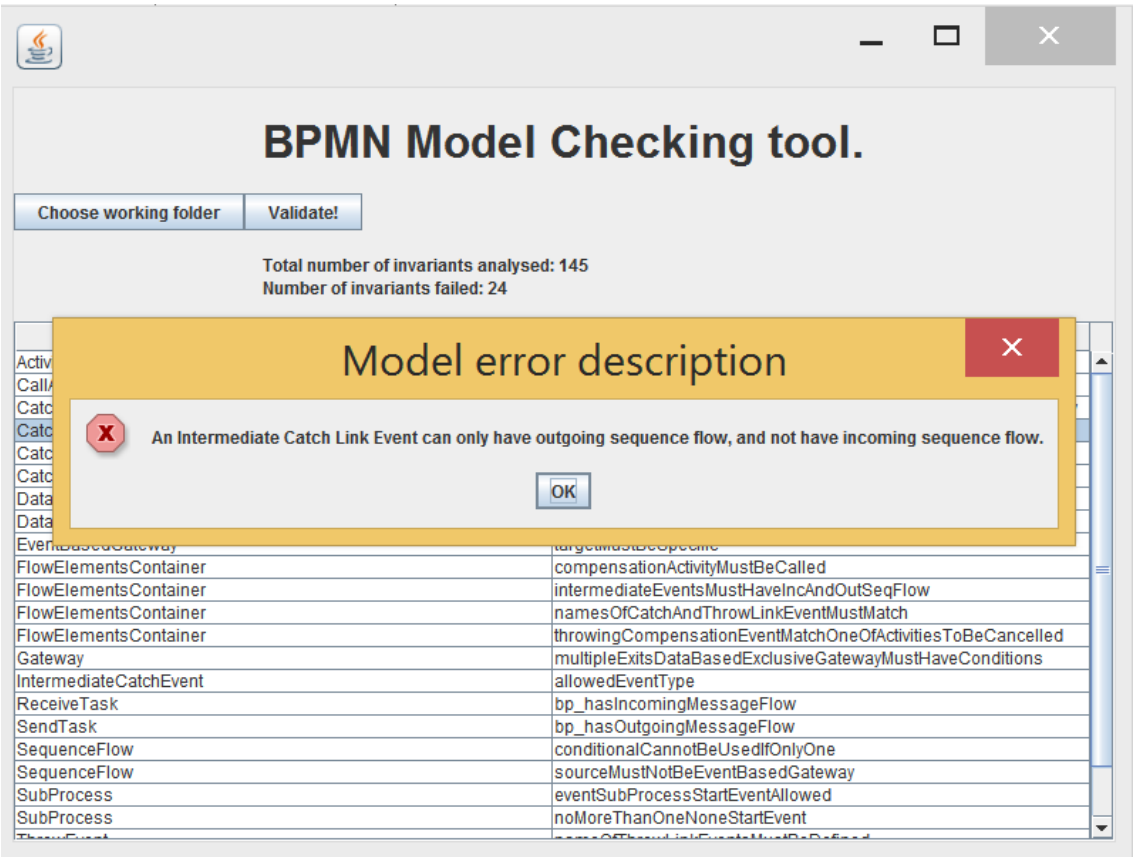


Fig. 3.9: Parser GUI - Invariant description.

3.6 Model checking architecture

A parser is a program to receive a specific input with a defined syntax, and is able to process it and use the content for a specific purpose. In our context, we need to construct a parser to read XPDL 2.2 content and understand all the XPDL 2.2 elements used, all the properties associated to each of its elements and even the relationship between different elements. Since the parser is able to read the XPDL content, we need that the parser generates USE input commands to allow create the elements, represented in the model, and according to the BPMN2 specification (M1). After the business process model designed using any BPMN2 modeling tool with functionality to export the model to XPDL format, the user need to export the model to XPDL format into one or several documents, and thereafter provide these files to the parser and start the model checking validation.

The USE environment is a system based on the Unified Modeling Language (UML) and the Object Constraint Language (OCL) that allows the user to specify and validate a specific information system. This informatin system is represented by a model, and the model is composed by classes and associations. On the other side, the user can specify OCL expressions in order to include additional integrity constraints to the model.

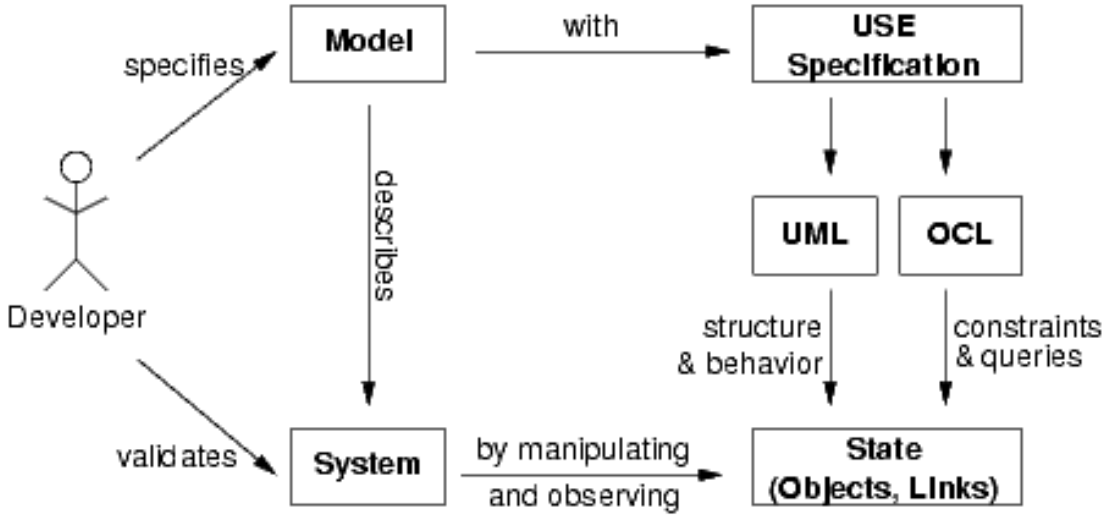


Fig. 3.10: USE approach - General view [7].

In figure 3.11 we have a components diagram explaining the context and architecture of the solution presented in this chapter. The BPMN Modeling Tool is an independent component that is used to produce the BPMN models and export these models to XPDL files. The parser has the responsibility to read the XPDL files content and produce an output file with USE commands that will instantiate, the BPMN model designed, in USE environment. The validator controller is the component responsible for consume the J-USE Api in order to execute USE commands to load the output and instantiate the BPMN model and commands to execute validations to the BPMN model and return the results. Thereafter, this component uses Swing technology in order to provide a user-friendly interface to the user and show the relevant results from the model validation. The USE is an independent component that receives USE commands to load the BPMN metamodel, to instantiate the BPMN model and to execute validations on BPMN model.

[7](source: USE Official Site)

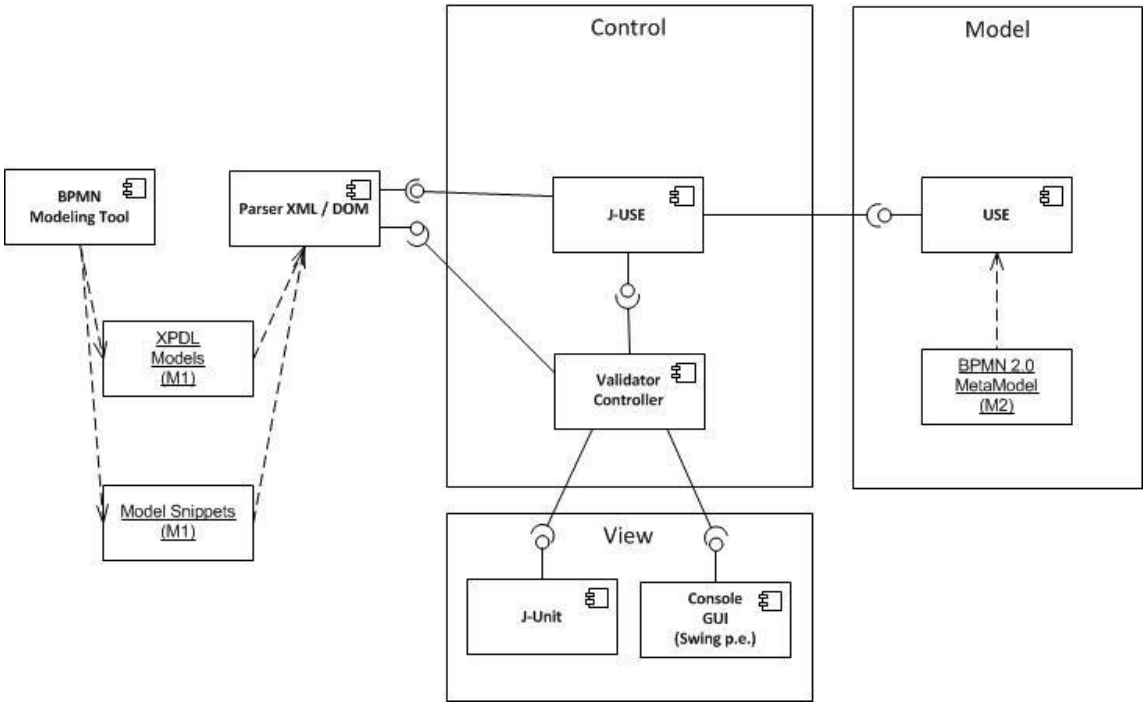


Fig. 3.11: BPMN Model Checkin Tool - Components Diagram.

3.7 XPDL to BPMN2 parser

Since we need to read the XPDL content exported from the BPMN models designed, we needed to develop a parser in order to interpret this content and identify each one of the elements included in this content.After some research, we found a library called JAXB. Basically, we obtained the XPDL 2.2 XSD (XML Schema Definition) located in [21] - http://www.xpdl.org/standards/xpdl-2.2/bpmnxpdl_40a.xsd, and we used the JAXB library to read this XSD file and generate the set of Java (programming language) classes for each one of the elements included in the XSD file. After generating the Java program with all classes, we have to change the program in order to be able to read files with XPDL content and instantiate the specific Java Class related with each of the elements detected in the XPDL content. On the other side, we have to change each one of the Java classes in order to generate the related USE command to create the BPMN2 element in the USE environment and fill its attributes and connections between other elements.

3.8 Integration of USE - parser - GUI

To allow adding model checking features to a BPMN modeling tool, such as the ones mentioned in the previous chapter, we have two options: (i) embedding the model checking features in the BPMN tool itself or, (ii) providing a separate model checking tool.

Table 3.2 summarize the pros and cons of each option.

Table 3.2: BPMN model checking features cenarios.

	Pros	**Cons**
(i) Embedded Model Checking Feature	Swifter user operation while performing a model check	BPMN tool must be open source or allow plugins / extensions; Integration is hard because it requires knowledge on the BPMN tool architecture; Integrations effort has to be reported for adding the Model checking to another BPMN tool
(ii) Independent Model Checking Tool	All BPMN tools, either open source or proprietary, supporting the same interoperability format, are eligible	Requires compatibility with model interoperability format, model checking is a two step process (i.e. model checking must be preceded by a model export)

After analysis of both cenarios, we decided to use the second scenario, where we focus on build an automated validation solution and easy to use, so the user can easily validate the BPMN models designed. This validation solution has USE source code integrated, the parser, the J-USE source code and required libs to use Swing technology. The USE source code is a requirement from J-USE, so we can run the USE environment and execute commands on it using the J-USE API.

[This page was intentionally left blank]

4. VALIDATION

In this chapter we present an experiment executed in order to test the metamodel-based validation facility. We used a set of thirty-nine (39) models produced in academy by students in the context of a business process modeling subject using BPMN. We describe the analysis performed on these models validation, to test the validation facility's effectiveness in model errors detection. Additionally, we validated the details in terms of elements instantiated to USE environment for each model, to guarantee that the parser is able to cover all BPMN constructs.

4.1 Coverage validation

Since we are using the XPDL format to export BPMN models, and use the XPDL files as input to the validation facility, we need to analyse the coverage of the parser in terms of mapping between the BPMN metamodel and the XPDL metamodel. This creates a dependency between BPMN and XPDL, especially in terms of mapping support between two metamodels (M2). In order to ensure the support and consequently guarantee that the parser is able to cover all BPMN metamodel constructs, we created a BPMN model sample with coverage of all the BPMN elements, so we can validate this model in the validation facility and check if it has the same number and type of elements created in the USE environment.

As annexes, we have the figures F.1, F.2, F.3 and F.4 that represent the model used for coverage analysis sake, where the main goal was to include all BPMN elements without worry on the model semantics.

In chapter F we have the number of meta-objects and metalinks that we obtained when we instantiate the BPMN metamodel in the USE environment. In table F.1 we represent the number of metaobjects instantiated for each of BPMN metaclass. In table F.2, we represent the number of metaobjects instantiated for each BPMN metaassociation. As we can see, some of the objects were not instantiated, so in table F.3 we have the list of not instantiated metaclasses and the corresponding rationale.

The tool used for this analysis was Bizagi Modeler, since, as we saw in a previous section, it is the most widely used tool nowadays. Apart of the limitations, we can see on

the tables results that most of the objects and associations were covered and we were able to instantiate and validate.

To summarize, after running the validation facility for the model in context, based on details shown, we can see the total list of different BPMN elements created, the total list of different links created and the corresponding cardinalities. Based on these details, we can see that the coverage test for this solution allowed us to provide some evidence that the validity threat associated to the lack of mapping between BPMN2 and XPDL 2.2 was mitigated.

4.2 Postmortem validation

Initially we used a set of models produced in academia from groups of students, in the context of a subject related with software process modeling with BPMN. We collected a set of thirty-nine (39) models, where each model was produced by a different group.

Each of these models was exported to XPDL and imported in the validation facility as described in previous chapter 3. We then checked the well-formedness and best practice rules, both expressed as OCL constraints, for each model and collected the information related with the validation of each of the invariants.

In table 4.1 we can see the set of models considered in this experiment. We decided to introduce the column *Model Reference* in order to have a short alias for each of the models and thus provide an improved readability of results.

Table 4.1: Set of models used in the experiment

Model Name	**Model Reference**
PL 1TP03 06_MDD 54430 60268 60988	M1
PLC1 01_XP 16220 30843 33395	M2
PLC1 02_XP 53864 54162 54179	M3
PLC1 03_XP 33390 37814 37827	M4
PLC1 04_XP 22127 33392	M5
PLC1 05_XP 35537 38076 38425	M6
PLC1 06_XP 54157 54172 55254	M7
PLC1 07_XP 54816 55726 54760	M8
PLC1 08_XP 26225 32110 33950	M9
PLC1 09_XP 30844 35529	M10
PLC1 10_XP 30823 35280	M11
TP02 01_XP 38438 38444	M12
TP02 02_XP 54774 54813	M13
TP02 03_XP 54181 54768 66432	M14
TP02 04_XP 28780 54761 54812	M15
TP02 06_XP 35270 54772 54779	M16
TP03 01_XP 54408 54378	M17
TP03 02_XP 37808 54407 54416	M18
TP03 03_XP 33573 38054 38057	M19
TP03 04_XP 54415 54411	M20
TP03 05_XP 38066 38077 38094	M21
TP03 06_XP 38065 38073 54403	M22
TP03 07_XP 38044 38085	M23
TP03 08_XP 23905 31722 35472	M24
TP03 09_XP 38051 28447	M25
TP04 01_XP 33572 35232 54379	M26
TP04 02_XP 38068 54419 55127	M27
TP04 03_XP 54388 54410 54429	M28
TP04 04_XP 40666 40929	M29
TP04 05_XP 38053 39903 55501	M30
TP04 06_XP 54386 54394 55653	M31
TP04 08_XP 54173 54783 55583	M32
TP05 01_XP 35278 37833 38471	M33
TP05 02_XP 35260 37822	M34
TP05 04_XP 30525 34307 54782	M35
TP05 06_XP 37997 38469 54771	M36
TP05 07_XP 25833 54777 54791	M37
TP05 09_XP 34329 38447 54778	M38
TP05 11_XP 54787 38477	M39

In table 4.2 we present the total number of invariants failed per model. The third column is related with model size, calculated as the sum of metamodel objects and links. The

fourth column is related with defect density, where it represents the quotient between number of invariants failed and model size. In other words, we are talking about ”defect density”. As average of model size we have eight hundred (800) objects, and as average number of invariants failed we have ten (10) invariants.

Table 4.2: Validation Facility Experiment - Total invariants failed per model

Model Reference	Number of invariants failed	Model Size	Defect Density
M1	9	956	0.0094
M2	10	972	0.0102
M3	9	935	0.0096
M4	7	599	0.0117
M5	6	396	0.0152
M6	6	1121	0.0054
M7	15	970	0.0165
M8	14	861	0.0163
M9	17	831	0.0205
M10	9	708	0.0127
M11	15	967	0.0155
M12	9	774	0.0116
M13	9	1175	0.0077
M14	9	632	0.0142
M15	10	848	0.0118
M16	10	1229	0.0081
M17	21	912	0.0230
M18	7	613	0.0114
M19	13	583	0.0223
M20	9	556	0.01619
M21	7	479	0.01461
M22	14	523	0.02677
M23	11	508	0.02165
M24	1	215	0.00465
M25	11	919	0.01197
M26	7	791	0.00885
M27	8	648	0.01235
M28	12	887	0.01353
M29	12	1972	0.006085
M30	7	469	0.01493
M31	4	326	0.01227
M32	16	676	0.02367
M33	7	609	0.01149
M34	13	1342	0.00969
M35	11	1058	0.01040
M36	11	879	0.01251
M37	9	726	0.01240
M38	8	618	0.01294
M39	10	920	0.01087

In figures 4.1 and 4.2, we present the analysis results of moderating relationship between number of invariants failed and model size, and, model size and defect density,

respectively. Pearson's Correlation was utilized to examine the correlation relationships between these variables, and it revealed that there is statistically significant correlation between number of invariants failed and model size, since $\rho = .012 < .05$ (see figure 4.1), and there is statistically significant correlation between model size and defect density, since $\rho = .019 < .05$ (see figure 4.2).

Correlations

		TotalFailedInvariants	ModelSize
TotalFailedInvariants	Pearson Correlation	1	.398*
	Sig. (2-tailed)		.012
	N	39	39
ModelSize	Pearson Correlation	.398*	1
	Sig. (2-tailed)	.012	
	N	39	39

*. Correlation is significant at the 0.05 level (2-tailed).

Fig. 4.1: Pearson Correlation between number of invariants failed and model size.

Correlations

		ModelSize	DefectDensity
ModelSize	Pearson Correlation	1	-.375*
	Sig. (2-tailed)		.019
	N	39	39
DefectDensity	Pearson Correlation	-.375*	1
	Sig. (2-tailed)	.019	
	N	39	39

*. Correlation is significant at the 0.05 level (2-tailed).

Fig. 4.2: Pearson Correlation between model size and defect density.

In figure C.1 we have the data collected from the experiment related with the validation of each invariant for each model used. The rows are related with the invariant id, and the columns are related with each model used for the experiment. Each row represents the validation result of an invariant for all the models used in the experiment.

Based on the results presented in this figure, we can conclude that the following invariants are the ones with the higher number of failures for all the models validated in the experiment.

Table 4.3: Validation Facility Experiment - Invariants with the higher number of failures

Id	Scope	Name	Fails	% failure
67	FlowNode	exclusivePathsMergingIntermedia-teByGateway	38	97%
33	DataStore	connectedDataAssociation	30	77%
56	FlowElementsContainer	explicitStartEvRequiresNoActiv-OrGatWithoutInSeqFlow	28	72%

The first row represents the invariant with *ID* equals to sixty-seven (67). This invariant belongs to the scope of *Flow Node* invariants. A *Flow Node*, or *Flow Object*, is the base element of *Event* element, *Activity* element and *Gateway* element, so all these elements inherit properties from *Flow Object* / *Flow Node*. In figure 4.3 we can see the BPMN2 elements hierarchy in order to understand the different inheritance relationships between BPMN2 elements.

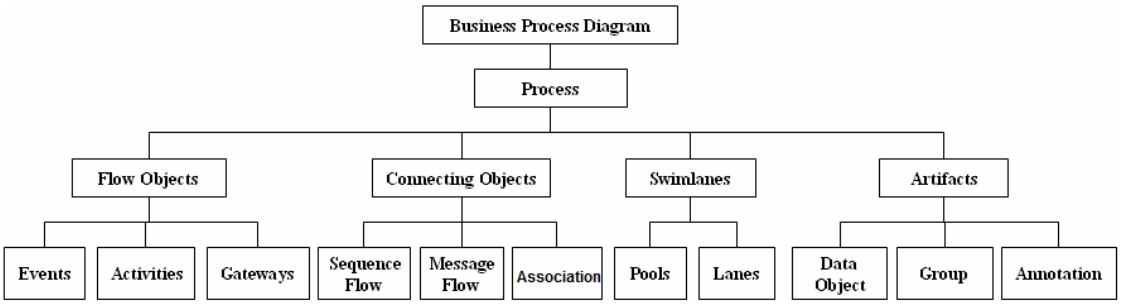


Fig. 4.3: BPMN2 Elements Hierarchy.

The invariant with *ID* equals to sixty-seven (67), represents a rule related with the outgoing sequence flows from a Exclusive Gateway, and it says that merging exclusive paths (outgoing sequence flows from Exclusive Gateway) requires a Gateway as a mediator, if another Gateway or an event follows. This rule is explained in figures C.6 and C.7.

The second invariant with *ID* equals to thirty-three (33) belongs to scope DataStore, and is related with Data Associations and Data Stores elements. This rule checks that a Data Object must have at least one connected Data Association. This rule is explained in figures C.8 and C.9.

The third invariant with *ID* equals to fifty-six (56), and belongs to scope of *Flow Elements Container*, and is related with the usage of explicit start events and/or end events. The rule guarantees that *Start Event* and *End Event* are optional. However, if there is at least one explicit *Start/End Event* in a container (*Process* or *SubProcess*), there must not

be other flow objects such as *Activity* or *Gateway*, without incoming/outgoing sequence flow. However there are some exceptions: *Compensation Activity* an *Event SubProcess* do not have incoming and outgoing *Sequence Flows*. This rule is explained in figures C.2 and C.3.

4.3 Field validation

In summary, this chapter presents a model checking experiment, where the model checking harness was partially developed in the work of [11] and extended in this dissertation. In terms of BPMN models validation, we conclude that this metamodel validation facility proved to be effective in models validation, since we used a set of models developed in *Bizagi* tool and no errors were found there, using Bizagi's validation system. However, we realised that these models contained several errors and we presented evidences of that. Additionally, these evidences allowed us to identify the most recurrent defects.

[This page was intentionally left blank]

5. STATISTICAL ANALYSIS

This chapter aims at proposing (i) reliability estimation to allow predict model defects in BPMN models based on model metrics, and (ii) analyse the influence of model's complexity on the model's fiability.

The model's complexity metrics were extracted using the MetaModel Driven Measurement (M2DM) approach and in context of [11] and were additionally defined in the BPMN2 metamodel specification. For more details related with the M2DM approach, see [11] and [23], whereas both works used this approach.

Then, we extracted model metrics from the USE environment during the validation of the models. These metrics represent model characteristics, e.g. number of a specific BPMN element occurrences in the model, model length, model difficulty, model modularization. In table D.1 we have the list of metrics specified and considered. Most of these metrics are related with number of occurrences in the model for a specific BPMN element, and consequently some of the metrics were omitted from the list, because their results were not relevant for the models considered, since the results were zero, or not zero for a small number of models, e.g. metrics for total number of *Complex Gateway* or *Text Annotation*. In the tables D.2 and D.3, we have the metrics results for each of models produced in Academia and considered to analysis.

The reliability data used for the aformentioned purposes consists in the results of the model checking data collection activity already described in chapter 4. We recall that these data relates to BPMN models produced either from students or professionals. The BPMN Model Complexity metrics were also collected alongside the reliability ones.

The results of model validation were grouped into two variables: the number of broken well-formedness rules and the number of broken best practice rules. Additionally, we created two more variables to store the defect density corresponding to these two variables, as already described in chapter 4. These four (4) variables define the set of dependent variables (DV) considered in this analysis, as described in table 5.1.

Table 5.1: Dependent variables (DV) analysis set

ID	Name	Description
BWFR	Broken Well-Formedness Rules (absolute)	Number of broken well-formedness rules in model's validation
BWFR_DENSITY	Broken Well-Formedness Rules (density)	Density of variable for number of broken well-formedness rules in model's validation
BBPR	Broken Best Practice Rules (absolute)	Number of broken best practice rules in model's validation
BBPR_DENSITY	Broken Best Practice Rules (density)	Density of variable for number of broken well-formedness rules in model's validation

Regarding the BPMN model complexity metrics, we considered the ones described in table 5.2:

Table 5.2: Independent variables (IV) analysis set

ID	Name
CFCM_RANK	Model's Control Flow Complexity - Rank
LENM_RANK	Model Length - Rank
VOLM_RANK	Model Volume - Rank
DIFFM_RANK	Model Difficulty - Rank
MODHKM_RANK	Model Modularization (Henry and Kafura Metric) - Rank

These variables are represented by absolute values, but since our analysis has the prerequisite of use descriptive variables as independent variables, we created a new descriptive variable for each of the metrics considered. Each of these descriptive variables is composed by a set of values, where these values represent levels in the variable. These values were defined based on percentils and measures in order to normalize the distribution of the absolute values across each of the levels. In figure 5.1 we have each of the values considered in each of these variables. The figure shows in first column the name of the IV, and the set of levels considered, the second column shows the label for each of the levels, and the third column shows the number of sample data included in each level.

Between-Subjects Factors

		Value Label	N
Model's Control Flow Complexity - Rank	.00	Simple	29
	1.00	Complex	29
Model Volume - Rank	.00	Small	14
	1.00	Medium	30
	2.00	Big	14
Model Length - Rank	1.00	Medium	43
	2.00	Large	15
Model Difficulty - Rank	.00	Easy	17
	1.00	Medium	27
	2.00	Hard	14
Model Modularization (Henry and Kafura Metric) - Rank	.00	Small	14
	1.00	Medium	30
	2.00	High	14

Fig. 5.1: Independent Variables Set Descriptives.

The tool used to these analysis was IBM SPSS Statistics version 23.

5.1 Model defects and Model metrics dependencies

In order to develop this analysis, we used the ANOVA statistical method. The ANOVA analysis is composed by main effects and interaction effects. A main effect represents the effect that the independent variable, also known as factor, has on the dependent vari- able, or outcome. The interaction represents the composed effect due to two or more independent variables in the dependent variable.

The set of independent variables are fixed effect factors, since the data has been gathered from all the levels of the factor that are ofinterest.

This analysis will have a X * Y between subjects design, whereas X equals the number of levels of the first independent variable and Y equals the number of levels of the second independent variable. The analysis is executed for each of the dependent variables, considering pairs of two independent variables for all the four independent variables considered.

In order to analyse in detail the effect of a IV on a DV at variable level's detail, we used the Compare Main Effects option for this analysis.

Additionally, we used Least Significant Difference for pairwise comparisons because all the factors have less than four levels.

In terms of plots, we used the factor with less number of levels as the separate lines value, and the factor with most number of levels as the horizontal axis value. In the plot, the Interaction is testing the hypothesis that the magnitude of the difference between levels of separate lines factor is equal across the levels of the horizontal axis factor.

Since we have unequal sample sizes for each level of the IVs (figure 5.1),we have to choose which is the most appropriate Model Type used in the ANOVA analysis. Based on [24] and [25], we have chosen the model type III and sums of square. This ANOVA model type III is an unweighted mean approach, so it is not going to take into account the fact that some groups have much larger sample sizes and weight those accordingly. Instead, it is going to consider that the sample sizes are equal and use a harmonic mean.

In terms of output results, we have the following outputs:

Tests of Between-Subjects Effects

This is the most important part of the output, since it tells us whether any of the independent variables was an effect on the dependent variable. The important data to look at in the table are the significance values of each of the independent variables.

Pairwise Comparisons

This output is used to determine which group differences are statistically significant for a specific factor.

Multiple Comparisons

This output is used to determine which group differences are statistically significant for a specific factor.

The difference between Pairwise and Multiple Comparisons is based on means used. The Pairwise Comparisons is based on estimated marginal means, while the Multiple Comparisons is based on observed means.

For each dependent variable, we have the following sets of factors:

- CFCM_Rank * DIFFM_Rank
- CFCM_Rank * LENM_Rank
- CFCM_Rank * MODHKM_Rank
- CFCM_Rank * VOLM_Rank
- DIFFM_RANK * MODHKM_Rank
- LENM_Rank * DIFFM_Rank

- LENM_Rank * MODHKM_Rank
- LENM_Rank * VOLM_RANK
- VOLM_Rank * DIFFM_Rank
- VOLM_Rank * MODHKM_Rank

Each set is composed by two independent variables, and represents an ANOVA analysis for the specific dependent variable.

For each ANOVA analysis, we collected and present here only significant results. If there is no results showed, means there was no significant results found, or those variables showed to have low explanatory power to the dependent variable.

The confidence interval considered for analysis is ninety-five (95%), and consequently σ equals five hundredths ($\sigma = 0.05$).

5.1.1 Broken Best Practice Rules - Absolute Value

There was no significant results found, or considerable explanatory power for all the factors considered to analysis and for the dependent variable in context.

5.1.2 Broken Best Practice Rules - Density

There was no significant results found, or considerable explanatory power for all the factors considered to analysis and for the dependent variable in context.

5.1.3 Broken Well-Formedness Rules - Absolute Value

5.1.3.1 CFCM _Rank * DIFFM_Rank

In next figure 5.2, there was a significant main effect of the Model Difficulty - Rank, on the total number of broken well-formedness rules - absolute value, with $F(2,52) = 11.285$, $\rho = 0.000 < 0.05$. And, there was a significant interaction of Model's Control Flow Complexity - Rank and Model Difficulty - Rank, with $F(2,52) = 4.057$, $\rho = 0.023 < 0.05$.

Tests of Between-Subjects Effects

Dependent Variable: Broken Well-Formedness Rules (absolute)

Source	Type III Sum of Squares	df	Mean Square	F	Sig.	Partial Eta Squared
Corrected Model	425.037[a]	5	85.007	9.175	.000	.469
Intercept	2500.173	1	2500.173	269.836	.000	.838
CFCM_RANK	.392	1	.392	.042	.838	.001
DIFFM_RANK	209.126	2	104.563	11.285	.000	.303
CFCM_RANK * DIFFM_RANK	75.184	2	37.592	4.057	.023	.135
Error	481.808	52	9.266			
Total	3791.000	58				
Corrected Total	906.845	57				

a. R Squared = .469 (Adjusted R Squared = .418)

Fig. 5.2: Tests of Between-Subjects Effects.

In next figure 5.3, we can see that there was a significant difference in Model Difficulty - Rank between Easy and Medium levels ($\rho = 0.003$), between Easy and Hard ($\rho = 0.000$) and between Medium and Hard levels ($\rho = 0.018$).

Pairwise Comparisons

Dependent Variable: Broken Well-Formedness Rules (absolute)

(I) Model Difficulty - Rank	(J) Model Difficulty - Rank	Mean Difference (I-J)	Std. Error	Sig.[b]	95% Confidence Interval for Difference[b]	
					Lower Bound	Upper Bound
Easy	Medium	-3.290*	1.051	.003	-5.399	-1.181
	Hard	-5.915*	1.252	.000	-8.428	-3.403
Medium	Easy	3.290*	1.051	.003	1.181	5.399
	Hard	-2.625*	1.076	.018	-4.785	-.465
Hard	Easy	5.915*	1.252	.000	3.403	8.428
	Medium	2.625*	1.076	.018	.465	4.785

Based on estimated marginal means

*. The mean difference is significant at the 0

b. Adjustment for multiple comparisons: Least Significant Difference (equivalent to no adjustments).

Fig. 5.3: Pairwise Comparisons of Model Modularization (Henry and Kafura Metric) - Rank.

In next figure 5.4, we had better ρ for all Model Difficulty - Rank levels, and there was a significant difference between Easy and Medium levels ($\rho = 0.000$), between Easy and Hard ($\rho = 0.000$) and between Medium and Hard levels ($\rho = 0.020$).

Multiple Comparisons

Dependent Variable: Broken Well-Formedness Rules (absolute)

LSD

(I) Model Difficulty - Rank	(J) Model Difficulty - Rank	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
					Lower Bound	Upper Bound
Easy	Medium	-4.078*	.9424	.000	-5.970	-2.187
	Hard	-6.483*	1.0986	.000	-8.688	-4.279
Medium	Easy	4.078*	.9424	.000	2.187	5.970
	Hard	-2.405*	1.0025	.020	-4.416	-.393
Hard	Easy	6.483*	1.0986	.000	4.279	8.688
	Medium	2.405*	1.0025	.020	.393	4.416

Based on observed means.
The error term is Mean Square(Error) = 9.266.

*. The mean difference is significant at the 0

Fig. 5.4: Multiple Comparisons of Model Difficulty - Rank.

In next figure 5.5, we can see that there was a significant difference between Model Difficulty - Rank and Model's Control Flow Complexity - Rank. For medium level of Model Difficulty, and between Simple and Complex levels of Model's Control Flow Complexity - Rank we have a significance of $\rho = 0.047 < 0.05$.

Pairwise Comparisons

Dependent Variable: Broken Well-Formedness Rules (absolute)

Model Difficulty - Rank	(I) Model's Control Flow Complexity - Rank	(J) Model's Control Flow Complexity - Rank	Mean Difference (I-J)	Std. Error	Sig.[b]	95% Confidence Interval for Difference[b]	
						Lower Bound	Upper Bound
Easy	Simple	Complex	-3.481	1.740	.051	-6.973	.012
	Complex	Simple	3.481	1.740	.051	-.012	6.973
Medium	Simple	Complex	2.400*	1.179	.047	.034	4.766
	Complex	Simple	-2.400*	1.179	.047	-4.766	-.034
Hard	Simple	Complex	1.650	1.801	.364	-1.964	5.264
	Complex	Simple	-1.650	1.801	.364	-5.264	1.964

Based on estimated marginal means

*. The mean difference is significant at the 0

b. Adjustment for multiple comparisons: Least Significant Difference (equivalent to no adjustments).

Fig. 5.5: Subjects Pairwise Comparisons.

In next figure 5.6, we can see there is statistically significant interaction for Model Difficulty - Rank equals Medium value, and in Model's Control Flow Complexity - Rank between levels Simple and Complex. As we can see, before the interaction Simple level of Model's Control Flow Complexity - Rank has less influence in Estimated Marginal Means than Complex level of Model's Control Flow Complexity - Rank, with the variance of Model Difficulty - Rank between Easy and Medium levels. After the interaction, the Simple level of Model's Control Flow Complexity - Rank has more influence on estimated marginal means than the Complex level.

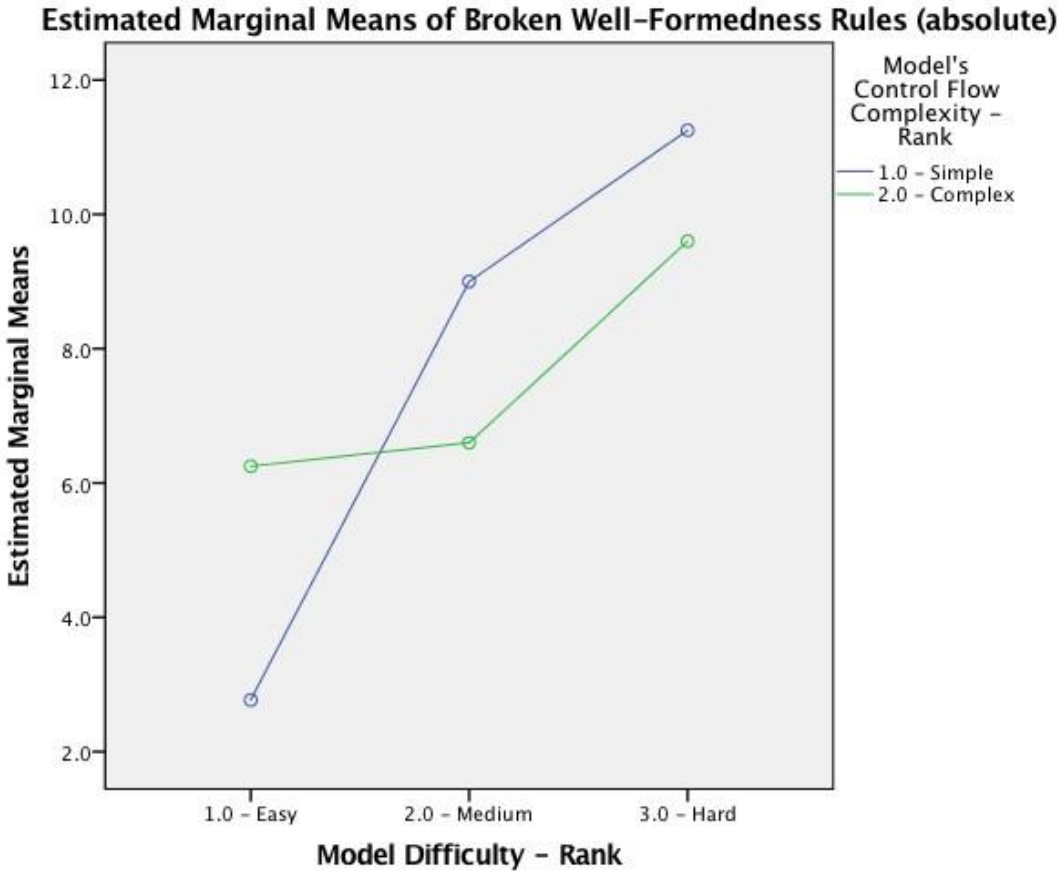


Fig. 5.6: Subjects Pairwise Comparisons.

5.1.3.2 DIFFM _RANK * MODHKM_Rank

In next figure 5.7, there was a significant main effect of Model Difficulty - Rank, with F(2,49) = 11.873, $\rho = 0.000 < 0.05$. Additionally, there was another significant main effect of Model Modularization (Henry and Kafura Metric) - Rank, with F(2,49) = 4.013, $\rho = 0.024 < 0.05$.

Tests of Between-Subjects Effects

Dependent Variable: Broken Well-Formedness Rules (absolute)

Source	Type III Sum of Squares	df	Mean Square	F	Sig.	Partial Eta Squared
Corrected Model	470.354[a]	8	58.794	6.600	.000	.519
Intercept	2473.337	1	2473.337	277.654	.000	.850
DIFFM_RANK	211.526	2	105.763	11.873	.000	.326
MODHKM_RANK	71.499	2	35.749	4.013	.024	.141
DIFFM_RANK * MODHKM_RANK	77.620	4	19.405	2.178	.085	.151
Error	436.490	49	8.908			
Total	3791.000	58				
Corrected Total	906.845	57				

a. R Squared = .519 (Adjusted R Squared = .440)

Fig. 5.7: Tests of Between-Subjects Effects.

In next figure 5.8, we can see that there was a significant difference in Model Modularization (Henry and Kafura Metric) - Rank between Medium and High levels (ρ=0.007).

Pairwise Comparisons

Dependent Variable: Broken Well-Formedness Rules (absolute)

(I) Model Modularization (Henry and Kafura Metric) - Rank	(J) Model Modularization (Henry and Kafura Metric) - Rank	Mean Difference (I-J)	Std. Error	Sig.[b]	95% Confidence Interval for Difference[b]	
					Lower Bound	Upper Bound
Small	Medium	.452	1.086	.679	-1.730	2.634
	High	-2.392	1.236	.059	-4.875	.091
Medium	Small	-.452	1.086	.679	-2.634	1.730
	High	-2.844*	1.019	.007	-4.891	-.798
High	Small	2.392	1.236	.059	-.091	4.875
	Medium	2.844*	1.019	.007	.798	4.891

Based on estimated marginal means

*. The mean difference is significant at the 0

b. Adjustment for multiple comparisons: Least Significant Difference (equivalent to no adjustments).

Fig. 5.8: Pairwise Comparisons of Model Modularization (Henry and Kafura Metric) - Rank.

In next figure 5.9, we can see that there was a significant difference in Model Difficulty - Rank between Easy and Medium levels (ρ=0.003), between Easy and Hard levels (ρ=0.000), and between Medium and Hard levels (ρ=0.024).

Pairwise Comparisons

Dependent Variable: Broken Well-Formedness Rules (absolute)

(I) Model Difficulty - Rank	(J) Model Difficulty - Rank	Mean Difference (I-J)	Std. Error	Sig.[b]	95% Confidence Interval for Difference[b]	
					Lower Bound	Upper Bound
Easy	Medium	-3.159*	1.019	.003	-5.206	-1.112
	Hard	-5.781*	1.203	.000	-8.198	-3.364
Medium	Easy	3.159*	1.019	.003	1.112	5.206
	Hard	-2.622*	1.122	.024	-4.877	-.367
Hard	Easy	5.781*	1.203	.000	3.364	8.198
	Medium	2.622*	1.122	.024	.367	4.877

Based on estimated marginal means

*. The mean difference is significant at the 0

b. Adjustment for multiple comparisons: Least Significant Difference (equivalent to no adjustments).

Fig. 5.9: Pairwise Comparisons of Model Difficulty - Rank.

In next figure 5.10, we can see that there was a significant difference in Model Modularization (Henry and Kafura Metric) - Rank between Small and High levels (ρ=0.001), and Medium and High levels (ρ=0.013).

Multiple Comparisons

Dependent Variable: Broken Well-Formedness Rules (absolute)

LSD

(I) Model Modularization (Henry and Kafura Metric) - Rank	(J) Model Modularization (Henry and Kafura Metric) - Rank	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
					Lower Bound	Upper Bound
Small	Medium	-1.443	.9660	.142	-3.384	.498
	High	-3.929*	1.1281	.001	-6.196	-1.662
Medium	Small	1.443	.9660	.142	-.498	3.384
	High	-2.486*	.9660	.013	-4.427	-.544
High	Small	3.929*	1.1281	.001	1.662	6.196
	Medium	2.486*	.9660	.013	.544	4.427

Based on observed means.
The error term is Mean Square(Error) = 8.908.

*. The mean difference is significant at the 0

Fig. 5.10: Multiple Comparisons of Model Modularization (Henry and Kafura Metric) - Rank.

In next figure 5.11, we can see that there was a significant difference in Model Difficulty - Rank between Easy and Medium levels (ρ=0.000), between Easy and Hard levels (ρ=0.000), and between Medium and Hard levels (ρ=0.018).

Multiple Comparisons

Dependent Variable: Broken Well-Formedness Rules (absolute)

LSD

(I) Model Difficulty - Rank	(J) Model Difficulty - Rank	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
					Lower Bound	Upper Bound
Easy	Medium	-4.078*	.9241	.000	-5.935	-2.221
	Hard	-6.483*	1.0772	.000	-8.648	-4.319
Medium	Easy	4.078*	.9241	.000	2.221	5.935
	Hard	-2.405*	.9830	.018	-4.380	-.429
Hard	Easy	6.483*	1.0772	.000	4.319	8.648
	Medium	2.405*	.9830	.018	.429	4.380

Based on observed means.
The error term is Mean Square(Error) = 8.908.

*. The mean difference is significant at the 0

Fig. 5.11: Multiple Comparisons of Model Difficulty - Rank.

5.1.3.3 VOLM _Rank * DIFFM_Rank

In next figure 5.12, there was a significant main effect of Model Difficulty - Rank, with $F(2,51) = 8.523$, $\rho = 0.001 < 0.05$. Additionally, we can see that there was a significant interaction between Model Volume - Rank and Model Difficulty - Rank, with $F(2,51) = 5.021$ and $\rho = 0.010 < 0.05$.

Tests of Between-Subjects Effects

Dependent Variable: Broken Well-Formedness Rules (absolute)

Source	Type III Sum of Squares	df	Mean Square	F	Sig.	Partial Eta Squared
Corrected Model	446.365[a]	6	74.394	8.239	.000	.492
Intercept	2404.854	1	2404.854	266.347	.000	.839
VOLM_RANK	2.907	2	1.453	.161	.852	.006
DIFFM_RANK	153.917	2	76.958	8.523	.001	.251
VOLM_RANK * DIFFM_RANK	90.665	2	45.332	5.021	.010	.165
Error	460.480	51	9.029			
Total	3791.000	58				
Corrected Total	906.845	57				

a. R Squared = .492 (Adjusted R Squared = .432)

Fig. 5.12: Tests of Between-Subjects Effects.

In next figure 5.13, we can see that there was a significant difference in Model Difficulty - Rank between Easy and Medium levels (ρ=0.000), and between Easy and Hard levels (ρ=0.000).

Pairwise Comparisons

Dependent Variable: Broken Well-Formedness Rules (absolute)

(I) Model Difficulty - Rank	(J) Model Difficulty - Rank	Mean Difference (I-J)	Std. Error	Sig.[d]	95% Confidence Interval for Difference[d]	
					Lower Bound	Upper Bound
Easy	Medium	-4.198[*,b]	1.081	.000	-6.369	-2.027
	Hard	-5.761[*,b,c]	1.133	.000	-8.036	-3.487
Medium	Easy	4.198[*,c]	1.081	.000	2.027	6.369
	Hard	-1.563[c]	1.136	.175	-3.843	.717
Hard	Easy	5.761[*,b,c]	1.133	.000	3.487	8.036
	Medium	1.563[b]	1.136	.175	-.717	3.843

Based on estimated marginal means

*. The mean difference is significant at the 0

b. An estimate of the modified population marginal mean (I).

c. An estimate of the modified population marginal mean (J).

d. Adjustment for multiple comparisons: Least Significant Difference (equivalent to no adjustments).

Fig. 5.13: Pairwise Comparisons of Model Difficulty - Rank.

In next figure 5.14, we can see that there was a significant difference between Model Length - Rank and Model Volume - Rank, for Small level of Model Difficulty - Rank, and between Small and Medium levels of Model Volume - Rank, with $\rho = 0.005$.

Pairwise Comparisons

Dependent Variable: Broken Well-Formedness Rules (absolute)

Model Difficulty - Rank	(I) Model Volume - Rank	(J) Model Volume - Rank	Mean Difference (I-J)	Std. Error	Sig.[d]	95% Confidence Interval for Difference[d]	
						Lower Bound	Upper Bound
Easy	Small	Medium	-4.500[*]	1.525	.005	-7.562	-1.438
		Big	.[b]	.	.	.	.
	Medium	Small	4.500[*]	1.525	.005	1.438	7.562
		Big	.[b]	.	.	.	.
	Big	Small	.[c]	.	.	.	.
		Medium	.[c]	.	.	.	.
Medium	Small	Medium	3.123	1.867	.100	-.625	6.871
		Big	2.533	2.194	.254	-1.872	6.939
	Medium	Small	-3.123	1.867	.100	-6.871	.625
		Big	-.589	1.510	.698	-3.622	2.443
	Big	Small	-2.533	2.194	.254	-6.939	1.872
		Medium	.589	1.510	.698	-2.443	3.622
Hard	Small	Medium	.[c]	.	.	.	.
		Big	.[c]	.	.	.	.
	Medium	Small	.[b]	.	.	.	.
		Big	-.422	1.676	.802	-3.787	2.943
	Big	Small	.[b]	.	.	.	.
		Medium	.422	1.676	.802	-2.943	3.787

Based on estimated marginal means

*. The mean difference is significant at the 0

b. The level combination of factors in (J) is not observed.

c. The level combination of factors in (I) is not observed.

d. Adjustment for multiple comparisons: Least Significant Difference (equivalent to no adjustments).

Fig. 5.14: Pairwise Comparisons of Model Volume - Rank and Model Difficulty - Rank.

In next figure 5.15, we can see that there was a significant difference in Model Difficulty - Rank between Easy and Medium levels (ρ=0.000), between Easy and Hard levels (ρ=0.000), and between Medium and Hard levels with $\rho = 0.019$.

Multiple Comparisons

Dependent Variable: Broken Well-Formedness Rules (absolute)

LSD

(I) Model Difficulty - Rank	(J) Model Difficulty - Rank	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
					Lower Bound	Upper Bound
Easy	Medium	-4.078*	.9303	.000	-5.946	-2.211
	Hard	-6.483*	1.0845	.000	-8.660	-4.306
Medium	Easy	4.078*	.9303	.000	2.211	5.946
	Hard	-2.405*	.9896	.019	-4.391	-.418
Hard	Easy	6.483*	1.0845	.000	4.306	8.660
	Medium	2.405*	.9896	.019	.418	4.391

Based on observed means.
The error term is Mean Square(Error) = 9.029.

*. The mean difference is significant at the 0

Fig. 5.15: Multiple Comparisons of Model Difficulty - Rank.

In next figure 5.16, we can see there is no statistically significant interaction between Model Difficulty - Rank and Model Volume - Rank, when Model Difficulty is equal Hard level, since there is no significant difference of estimated marginal means across Medium and Big levels of Model Volume - Rank. On the other side, for Small level of Model Volume - Rank, we can see there is significant difference of estimated marginal means between Easy and Medium levels of Model Difficulty - Rank.

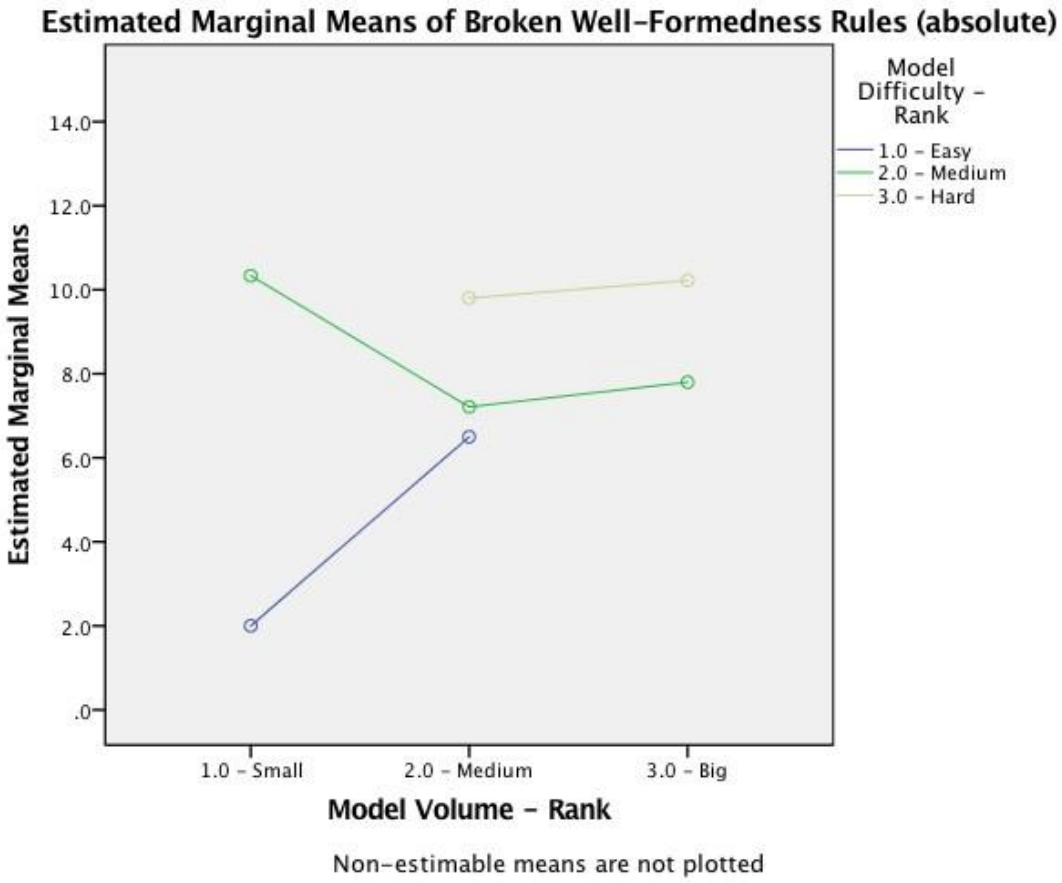


Fig. 5.16: Broken Well-formedness Rules - Absolute Value - Model Volume - Rank and Model Difficulty - Rank Plot.

5.1.3.4 Conclusions

Based on previous results shown, we can conclude that the Model Difficulty - Rank influences in the number of model defects in category Well-formedness rules in all its levels. On the other side, for the models with Small level of Model Modularization (Henry and Kafura Metric) - Rank, the metric Model's Control Flow Complexity - Rank showed to has high influence in value of estimated marginal means. In terms of Model Volume - Rank, we saw that there was significant difference on estimated marginal means when comparing models with Small level and models with Medium or Big level.

5.1.4 Broken Well-Formedness Rules - Density

There was no significant results found, or considerable explanatory power for all the factors considered to analysis and for the dependent variable in context.

5.1.5 Human factor

Based on previous results presented, we can conclude that the set of model metrics considered are not statistical significant to estimate the model smells considered as dependent variables. And, apart from the fact that some interactions between these metrics showed to be statistical significant, they showed to have low explanatory power to explain the dependent variables, where the best explanatory power reached was near *R squared* ≈ 0.5 (50%). Based on these facts, we decided to introduce the human factor, and analyse the effect of these metrics considering the human factor that indicates if the models were produced by students or by professionals.

In figure 5.17, we can see the descriptive statistics for the factors Origin - Rank and Model Complexity - Rank.

Descriptive Statistics

Dependent Variable: Broken Well-Formedness Rules (absolute)

Model Origin	Model's Control Flow Complexity – Rank	Mean	Std. Deviation	N
Professional	Simple	2.636	1.9633	11
	Complex	4.250	1.2817	8
	Total	3.316	1.8575	19
Students	Simple	8.889	4.4838	18
	Complex	8.857	2.3299	21
	Total	8.872	3.4426	39
Total	Simple	6.517	4.8079	29
	Complex	7.586	2.9462	29
	Total	7.052	3.9887	58

Fig. 5.17: Broken Well-formedness Rules - Absolute Value - Origin - Rank and Model Complexity - Rank - Descriptive Statistics.

In next figure 5.18, there was a significant main effect of Origin - Rank, with $F(1,54) = 39.885$, $\rho = 0.000 < 0.05$.

Tests of Between-Subjects Effects

Dependent Variable: Broken Well-Formedness Rules (absolute)

Source	Type III Sum of Squares	df	Mean Square	F	Sig.
Corrected Model	406.450[a]	3	135.483	14.621	.000
Intercept	1901.554	1	1901.554	205.206	.000
Origin	369.597	1	369.597	39.885	.000
CFCM_RANK	7.842	1	7.842	.846	.362
Origin * CFCM_RANK	8.485	1	8.485	.916	.343
Error	500.395	54	9.267		
Total	3791.000	58			
Corrected Total	906.845	57			

a. R Squared = .448 (Adjusted R Squared = .418)

Fig. 5.18: Broken Well-formedness Rules - Absolute Value - Origin - Rank and Model Complexity - Rank - Tests of Between-Subjects Effects.

In next figure 5.19, we can see that there is a statistically significant interaction between Origin - Rank and Model Complexity - Rank, when the Origin - Rank is equal to ”2 - Students”. Additionally, we can see in the figure that there is influence from the factor Origin - Rank in the dependent variable Broken Well-Formedness Rules (absolute), whereas there is a significant increase in the estimated marginal means of the dependent variable, comparing the origin *Professionals* models with the origin of *Students* models.

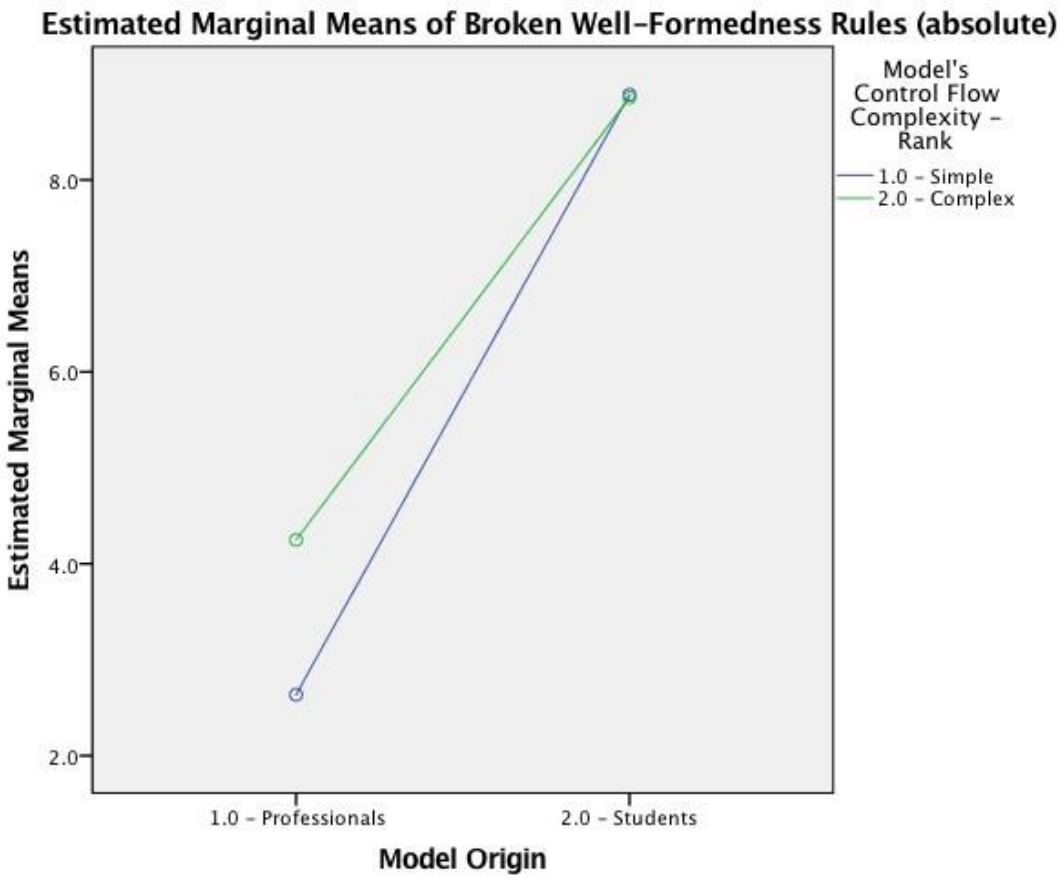


Fig. 5.19: Broken Well-formedness Rules - Absolute Value - Origin - Rank and Model Complexity - Rank - Interactions Plot.

[This page was intentionally left blank]

6. RELATED WORK

6.1 Introduction

In this section we describe the related work, classified according to a proposed taxonomy. The selected set of papers describing the related work were chosen based on a continuous search process, where the queries were adjusted based on previous results obtained. Initially, we searched for related work using the following string: "Validation Business Process Model" + "BPMN". From that research, only a few results were found, since there is no much relevant work done based on validating BPMN models. As so, we made an adjustment in our search query, omitting the reference to BPMN. Therefore, obtaining more results. With these new findings, we constructed a list of works related with the subject and relevant to our research. Hereafter, we defined the taxonomy to classify the strengths and weaknesses of each work in order to help us to compare the different proposals. This taxonomy is composed by a set of criteria that represents aspects that all the related work has in common. For each criterion an ordinal scale was defined to allow classify each work.

6.2 Taxonomy

•VALIDATION

This criterion intends to evaluate if the proposed approach in the related work is applied in practice based on a illustrative example or an empirical study. This criterion is fundamental to understand if the proposed approach can be applied in practice or not. This criterion is classified based on the following categories:

- D – Non validation done: There was no validation done in practice.
- C – Just illustrative example: There was done only an illustrative example in practice.

- B – Sample without statistical results presented: There was a statistical validation with a considerable sample, but the statistical results are not presented in the published work.
- A – Sample with statistical results presented: There was a statistical validation with a considerable sample and the statistical results are presented in the published work.

•**REPLICABILITY**

This criterion refers to the ability of an entire experiment or study to be replicable, or by someone else working independently. It is one of the main principles of the scientific method. The result values are said to be commensurate if they are obtained (in distinct experimental trials) according to the same replicable experimental description and procedure. This criterion is classified based on the following categories:

- D – Not replicable: The experiment or study developed in related work can not be replicable since there is not enough information to reproduce the procedure did before.
- C – Partially replicable: The experiment or study developed in related work can be partially replicable since there is partially information to replicate what the procedure did before, and the results obtained were not presented.
- B – Largely replicable: The experiment or study developed in related work can be largely replicable since there is all information to replicate the procedure did before, but the results obtained were not presented.
- A – Fully replicable: The experiment or study developed in related work is fully replicable since there is all information to replicate the procedure did before, and the results obtained previously.

•**AUTOMATIZATION**

This criterion characterizes the ability to apply the proposed model validation technique without human intervention. This criterion is classified based on the following categories:

- C - Not automated: The proposed model validation technique can be applied only with human intervention.

- B - Partially automated: The proposed model validation technique can be applied with some automated steps and some steps with human intervention.
- A - Automated: The proposed model validation technique can be applied without human intervention.

•**EXTENSIBILITY**

This criterion classifies the ability to extend the solution in adding an additional model smell to the set of detected ones and the level of effort required to implement the extension.

- C – Unknown: It is not known or mentioned by the author if it is possible to extend the proposed solution.
- B – Hard to extend: It is possible to extend the proposed solution, but it requires a lot of effort.
- A – Easy to extend: It is possible to extend the proposed solution and it requires little effort.

•**APPLICABILITY**

This criterion classifies if the proposed approach is a valid solution to solve the problem in its current state of the art. This criterion is classified based on the following categories:

- C – Not applicable: The proposed solution is not applicable in the current state of art since the problem does not exist anymore, or was solved by another better solution.
- B – Partially applicable: The solution proposed in related work only solves a part of the related problem.
- A – Applicable: The solution proposed in related work can solve the related problem that still exists in the current state of the art.

6.3 Revision of related work

6.3.1M. Chinosi, "Representing Business Processes: Conceptual Model and Design Methodology" - [1]

Objective: Implement a metamodel based facility to overcome the weak points of BPMN 1.1 specifications.

Abstract: In this article, a new conceptual model for BPMN was developed from scratch, with a clear metamodel and its related XML-based serialization which might provide a self validating mechanism for checking BP syntax and semantics. To complete it, was defined a set of syntax and semantic rules to improve validation system of BPMs and overcome the weak points of BPMN 1.1 specification.

Critique: Most of the weak points of BPMN 1.1 specification were overcome with the official BPMN 2.0 specification published. On the another hand, the solution was developed from scratch. Since their intention was to improve a validation system of BPMs based on BPMN, from which they could use the existing work done already by OMG on BPMN 1.1 specification and develop the validator with the considered rules indepen- dent of BPMN metamodel. On the positive side, the solution presented allows to make a real-time validation of a model (while the designer is constructing it) and the validation may be totally automatized. Finally, an illustrative example with the proposed solution is presented but a statistical validation, using a considerable sample, was not performed.

Criterion	Classification
Validation	C
Replicability	C
Automatization	A
Extensibility	A
Applicability	C

6.3.2J. Mendling, G. Neumann, and W. Aalst, "Understanding the Occurrence of Errors in Process Models Based on Metrics" – [2]

Objective: Discuss the theoretical connection between errors and metrics, and provide a comprehensive validation based on an extensive sample of EPC process models from practice.

Abstract: In this article, it was analyzed the relation between formal errors and a set of metrics that capture various structural and behavioral aspects of a process model for predicting the former. A comprehensive validation, based on an extensive sample of EPC process models from practice, was provided.

Critique: The authors claim that existing proposals for predicting errors in process

models lack empirical validation. Their validation experiment is a good step towards mitigating that problem. On the other hand, since collecting new metrics require statistical adjustment, the solution proposed becomes hard to maintain and extend.

Criterion	Classification
Validation	A
Replicability	A
Automatization	B
Extensibility	B
Applicability	A

6.3.3A. Ahmed, D. Gero, and W. Mathias, "Efficient Compliance Checking Using BPMN-Q and Temporal Logic" – [3]

Objective: This paper presents an approach for compliance checking for BPMN process models using BPMN-Q queries.

Abstract: Compliance rules describe regulations, policies and quality constraints business processes must adhere to. Given the large number of rules and their frequency of change, manual compliance checking can become a time-consuming task. Because of that, there was the need to develop an automated solution for compliance checking in BPMN process models. In this paper, this approach was developed using BPMN-Q as a query language, since it helps identify the set of process models that are subject to compliance checking.

Critique: One of the negative points of the proposed solution is the fact that the described version does not provide detailed information in case of non-compliance process models. One positive point is the capacity of this approach to be applied to any process modeling language, besides the BPMN modeling language used in this experiment.

Criterion	Classification
Validation	C
Replicability	A
Automatization	A
Extensibility	B
Applicability	A

6.3.4Kluza, K., Nalepa, G. J., Szpyrka, M., & Ligeza, A. (2011). "Proposal of a Hierarchical Approach to Formal Verification of BPMN Models Using Alvis and XTT2 Methods" – [4]

Objective: Proposes a new approach to formal verification of BPMN models using the Alvis modeling language and the XTT2 knowledge representation.

Abstract: The paper presents preliminary results of the research concerning verification of BPMN models. An approach is proposed using the Alvis modeling language [26] for the global verification of the model structure and the XTT2 knowledge representation [27] for the local verification i.e. verification of single BPMN elements in the model. The structure of the BPMN model can be analyzed using a translation to an Alvis model. These models can be verified with dedicated tools, and their properties can be linked to the properties of the original BPMN model. On the other hand, selected BPMN elements can be verified using the XTT2 decision tables. Several BPMN elements can be translated to XTT2 and checked using the HeaRT rule engine [28] with the HalVA [29] verification and analysis tool.

Critique: As negative point, the evaluation process is performed upon an illustrative example with a simple model. Because of that, it is not so easy to the author to get conclusions of solution's validation, and consequently the quality of those is poor. On the other side, this approach, does not support OR-join gateways, and has limitations with lack of support of the multiple merge and split elements.

Criterion	Classification
Validation	C
Replicability	C
Automatization	B
Extensibility	B
Applicability	A

6.4 Results and Conclusions

To summarize, we realized that there is much more relevant work done related with validation of BPMs based on other modeling languages (EPCs, Petri Nets, rather than BPMN). Usually, on their work it is mentioned the technique addressed to implement the solution with the aim of improving the validation system of BPM modeling tools. We realized that

in related work that we found, the most common technique is based on metrics that seem to be related with the probability that a specific error occurs during the modeling phase. In other words, we found more work on defect prediction than on defect detection.

Secondly, based on related work, we may conclude that each of the approaches reviewed for BPM validation, independently of the modeling language, can be classified in one of the following categories:

- Implementation of a complete conceptual model and respective metamodel from the scratch.
- Solutions based on conversion of BPMN models to another type of modeling languages and then the usage of validation tools already developed.
- Solutions based on metrics, where the latter are used to predict errors introduced by designers during the modeling phase.

In table 6.1 we have the list of related work considered into analysis. In table 6.2, we have the final related work evaluation based on the criterion specified previously. Additionally, it was included the evaluation of the validation facility (P5) presented in context of this work and [11], in order to provide a comparison between all the proposals considered in this chapter.

Table 6.1: Related Work List

Paper Id	Paper Name
P1	Chinosi2009
P2	Mendling, et al., COOPIS, 2007
P3	Awad, et al., 2008
P4	Kluza, et al., 2011

Table 6.2: Related Work Criterion Evaluation

Paper Id	Validation	Replicability	Automatization	Extensibility	Applicability
P1	C	C	A	A	C
P2	A	A	B	B	A
P3	C	A	A	B	A
P4	C	C	B	B	A
P5	A	A	A	A	A

Besides of the type of approach followed, there might be some common limitations in a few solutions. A part of them might not have a real-time validation system, and

usually the validation is made with a human interaction way from the designer, which might explain the reason why the validation system is independent of the modeling tool and why there might be no integration between them. For the solutions based on metrics, the matter may need a considerable effort, necessary to extend the solution, in particular, to add new rules for validation system. This effort, for every metric, is based on creating a sample with models examples, some with errors introduced and some without, in a way to extract metrics from this sample and use them on creation of a new estimation model. Also, other intermediate tasks may be done in order to get valid measures for metrics.

Finally, based on classifications obtained from criteria, we might notice that a large amount of works studied in context of state-of-art may not have a consistent validation of results, probably because the source could be too theoretic or it was made as a sample illustrative example. In the case of reusability, almost all (except [1]) works studied were classified as 'fully reusable', which could mean that there could be evidence and information of all processes executed in context of the research, and consequently it might be reusable in future works. Relatively to replicability, only [2] may present an ample data necessary to replicate all the study done and reproduce results in way to confirm veracity of results presented in work. In the case of a usual automation, the solutions might be full or partially automated. In the case of extensibility, almost all solutions (except [1]) tend to be difficult to extend and a set of steps would be needed to implement the extension. In case of applicability, almost all solutions (except [1]) may be applicable to the current state-of-art.

[This page was intentionally left blank]

7. CONCLUSIONS AND FUTURE WORK

7.1 Conclusions

Summarizing, this MSc dissertation represents a continuation of a PhD research work produced at the QUASAR research group [11] and took as input two of its deliverables:

(i) A catalogue of BPMN2 model smells [12]

(ii) A specification of the BPMN2 Metamodel, based on OMG specification [10], expressed as a UML class diagram, enriched with OCL constraints representing Well-formedness and best practice rules (in modeling business process models).

Based on the aforementioned technical report we carried out a tool survey in order to characterize in detail the current state of practice regarding the detection of design errors in BPMN2 modeling tools. With this survey we concluded that there are serious flows in the implementation of validation systems of each of the modeling tools considered. Then, we developed a validation facility that is able to read a BPMN model expressed in XPDL content, interpret all the content and identify BPMN objects (elements and associations) included in the model. Thereafter, it generates USE commands in order to instantiate each of these objects to USE environment. Additionally, in this facility we used the J- USE api in order to have integration with USE environment, so we could instantiate the provided BPMN model and execute the set of invariants defined in the BPMN metamodel deliverable. Finally, this tool is able to identify modeling errors and show the relevant information to the user. Thereafter, we developed an experiment using the validation facility that allowed us to validate a set of BPMN2 models developed in academy by groups of students in the context of a Business Process Modelling subject. We concluded in this experiment that the validation facility is able to find errors that the modeling tool used to design initially these models was not. On the other hand, we could analyse the data collected and see what were the most recurrent failed rules and the less recurrent failed rules in these models validation.

Based on these results, we concluded that this validation facility is an added value since it allows preventing the modeler from producing flawed models, while it provides a self-

learning experience, since the rationale for each detected model smell is provided. In the medium to long term, we expect modelers to produce less and less BPMN model smells, due to the induced learning effect.

7.2 Future Work

Apart from the fact of the solution presented and used in the experiments in this dissertation, there is several steps that could be considered in order to improve the quality of the solution, and consequently to improve the results for the users that will use this solution. To start, the BPMN metamodel specification is not finished in terms of invariants specification, since there is some invariants nowadays that are not being used in the metamodel. Additionally, the error description messages showed in validation process are not specified for all the invariants evaluated, so there is the need to change the BPMN metamodel to provide simple and clear messages for each of the invariants evaluated, in order to allow the user to understand what is the existing error and how can fix it. Relatively to the parser functionality, as mentioned before, some of the elements were not possible to test its coverage, since we had some tool's limitations. One possible solution would be to consider using other tools, one or more, that would allow to reproduce models with the specific elements to test the coverage. Finally, the validation facility could be extended in order to register the historic of modeling errors introduced by the user, and provide the functionality to the user generate reports with this information and understand in which BPMN scopes he introduces more errors and in which type of BPMN elements, so the user could improve his knowledge in this areas and improve his BPMN modeling skills and potentially avoid introduce these errors in future.

[This page was intentionally left blank]