

### Lecture Notes in Computer Science

The LNCS series reports state-of-the-art results in computer science research, development, and education, at a high level and in both printed and electronic form. Enjoying tight cooperation with the R&D community, with numerous individuals, as well as with prestigious organizations and societies, LNCS has grown into the most comprehensive computer science research forum available.

The scope of LNCS, including its subseries LNAI and LNBI, spans the whole range of computer science and information technology including interdisciplinary topics in a variety of application fields. The type of material published traditionally includes

- proceedings (published in time for the respective conference)
- post-proceedings (consisting of thoroughly revised final full papers)
- research monographs (which may be based on outstanding PhD work, research projects, technical reports, etc.)

... have been added featuring  
... value components; these  
... of lectures given at  
... mediated coverage  
... broader community  
... published electronically

ISBN 978-3-642-05414-3



9 783642 105414 >

Lecture Notes in  
Computer Science

LNCS

LNAI

LNBI

Abran et al. (Eds.)



LNCS  
5891

Software Process and Product  
Measurement

LNCS 5891

International Conferences IWSM 2009 and Mensura 2009  
Amsterdam, The Netherlands, November 2009  
Proceedings

 Springer

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

- David Hutchison  
*Lancaster University, UK*
- Takeo Kanade  
*Carnegie Mellon University, Pittsburgh, PA, USA*
- Josef Kittler  
*University of Surrey, Guildford, UK*
- Jon M. Kleinberg  
*Cornell University, Ithaca, NY, USA*
- Alfred Kobsa  
*University of California, Irvine, CA, USA*
- Friedemann Mattern  
*ETH Zurich, Switzerland*
- John C. Mitchell  
*Stanford University, CA, USA*
- Moni Naor  
*Weizmann Institute of Science, Rehovot, Israel*
- Oscar Nierstrasz  
*University of Bern, Switzerland*
- C. Pandu Rangan  
*Indian Institute of Technology, Madras, India*
- Bernhard Steffen  
*University of Dortmund, Germany*
- Madhu Sudan  
*Microsoft Research, Cambridge, MA, USA*
- Demetri Terzopoulos  
*University of California, Los Angeles, CA, USA*
- Doug Tygar  
*University of California, Berkeley, CA, USA*
- Gerhard Weikum  
*Max-Planck Institute of Computer Science, Saarbruecken, Germany*

Alain Abran René Braungarten  
Reiner R. Dumke Juan J. Cuadrado-Gallego  
Jacob Brunekreef (Eds.)

# Software Process and Product Measurement

International Conferences IWSM 2009 and Mensura 2009  
Amsterdam, The Netherlands, November 4-6, 2009  
Proceedings

 Springer

## Preface

Since 1990 the International Workshop on Software Measurement (IWSM) has been held annually and is now in its 19<sup>th</sup> edition. The International Conference on Software Process and Product Measurement (Mensura) was initiated in 2006 and is now in its third edition. The editions of IWSM/Mensura have been combined since 2007 to foster research, practice and exchange of experiences and best practices in software processes and products measurement. The 2009 editions were held during November 4–6, 2009 in Amsterdam, organized jointly with The Netherlands Association for Software Measurement (NESMA)<sup>1</sup> and kindly hosted by Hogeschool van Amsterdam<sup>2</sup>.

Today the pressure for more efficient software development processes delivering appropriate quality is constantly increasing. But who knows how efficient one's own current development process actually is and whether the quality of delivered products is really appropriate? Did we substantially improve with all the improvement effort spent? How can we answer all these questions if not by measuring both software processes and software products?

Software measurement is a key technology with which to manage and to control software development projects. Measurement is essential of any engineering activity, by increasing the scientific and technical knowledge for both the practice of software development and for empirical research in software technology. IWSM/MENSURA facilitates the exchange of software measurement experiences between theory and practice.

Software process evaluation and improvement require quantified methods and technologies. Issues such as the applicability of measures and metrics to software, the efficiency of measurement programs in industry and the theoretical foundations of software engineering have been researched in order to evaluate and improve modern software development approaches.

These proceedings are testimonies of many of the software measurement concepts developed and of their related use in industry. These proceedings are of particular interest to software engineering researchers, as well as to practitioners, in the areas of project management and quality improvement programs, for both software development and software maintenance.

This volume comprises the proceedings of IWSM/Mensura 2009 and consists of the final papers presented at these joint events. Each one of these papers has been thoroughly refereed and extended in order to be accepted for publication. The IWSM/Mensura Steering Committee is proud to have — once more — obtained the approval of Springer to publish this third edition of the joint conference proceedings in the prestigious *Lecture Notes in Computer Science*:

<sup>1</sup> <http://www.nesma.nl/>

<sup>2</sup> <http://www.hva.nl/>

### Volume Editors

Alain Abran  
Université du Québec  
1100, rue Notre-Dame Ouest  
Montréal, Québec, Canada H3C 1K3  
E-mail: Alain.Abran@csmtl.ca

René Braungarten  
Bosch Rexroth Electric Drives and Controls GmbH  
Bürgermeister-Dr.-Nebel Str. 2  
97816 Lohr am Main, Germany  
E-mail: rene.braungarten@boschrexroth.de

Reiner R. Dumke  
Oto-von-Guertcke-Universität Magdeburg  
Universitätsplatz 2  
39106 Magdeburg, Germany  
E-mail: dumke@ivs.cs.uni-magdeburg.de

Juan J. Cuadrado-Gallego  
Universidad de Alcalá  
O24, Autovía A2, Km. 31.7  
28805 - Alcalá de Henares, Madrid, Spain  
E-mail: jjcg@uah.es

Jacob Brunekeerf  
University of Applied Science HvA  
Weesperzijde 190  
1097 DZ Amsterdam, The Netherlands  
E-mail: j.j.brunekreef@hva.nl

Library of Congress Control Number: 2009937647  
CR Subject Classification (1998): D.2, D.2.1, D.2.8, D.4.8, F.1.3, D.2.9  
LNCS Sublibrary: SL 2 – Programming and Software Engineering  
ISSN 0302-9743  
ISBN-10 3-642-05414-5 Springer Berlin Heidelberg New York  
ISBN-13 978-3-642-05414-3 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.  
springer.com

© Springer, Verlag Berlin Heidelberg 2009  
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India  
Printed on acid-free paper SPIN: 12791026 06/3180 5 4 3 2 1 0

(LNCS) series. We hope to maintain this collaboration for the future editions of these joint events.

We wish to express our gratitude to the sponsors of the IWSM / Mensura 2009 for their essential contribution to the conference. We also wish to express our gratitude to the organizers of IWSM / Mensura 2009 for their tireless dedication.

November 2009

Alain Abran  
René Braumgarten  
Reiner R. Dumke  
Juan J. Cuadrado-Gallego  
Jacob Brunekreef

## Organization

### General Chairs

Alain Abran  
University of Québec / ÉTS, Montréal (Québec), Canada  
Reiner R. Dumke  
Otto-von-Guericke-University, Magdeburg, Germany  
Juan J. Cuadrado-Gallego  
University of Alcalá, Madrid, Spain  
Jacob Brunekreef  
Amsterdam University of Applied Sciences, Amsterdam, The Netherlands

### Organization Chair

Jacob Brunekreef  
Amsterdam University of Applied Sciences, Amsterdam, The Netherlands

### Proceedings Chair

René Braumgarten  
Bosch Rexroth Electric Drives and Controls GmbH, Lohr am Main, Germany

### Program Committee Chair

Reiner R. Dumke  
Otto von Guericke University, Magdeburg, Germany

### Program Committee Members

Rafa Al Qutaish	Applied Science University Amman, Jordan
Luigi Buglione	Engineering.IT S.p.A., Italy
François Coallier	ÉTS, Montréal (Québec), Canada
Darren Dalcher	National Centre for Project Management, Middlesex University, UK
Ton Dekkers	Galorath International Ltd., UK
Jean-Marc Desharnais	ÉTS, Montréal (Québec), Canada
Axel Dold	Daimler AG, Sindelfingen, Germany
María J. Domínguez-Alda	University of Alcalá, Spain
Christof Ebert	Vector Consulting, Stuttgart, Germany
Marian Fernández de Sevilla	University of Alcalá, Spain
Bernd Gebhard	BMW AG, Munich, Germany

VIII Organization

- Marcela Genero  
 Najj Habra  
 Nadine Hanchutte  
 Hans-Georg Hoopf  
 Ali Idri  
 Taghi M. Khoshgoftaar  
 Claus Lewerentz  
 Marek Leszak  
 Peter Liggesmeyer  
 Mathias Lothar  
 Roberto Meli  
 Dirk Meyerhoff  
 Enriqueta Muel  
 Jürgen Münch  
 Olga Ormandjieva  
 Oscar Pastor  
 Frances Paulisch  
 Luca Santillo  
 Andreas Schmiestendorf  
 Asma Sellami  
 Harry Sneed  
 Charles Symons  
 Manar Abu Talib  
 Hannu Toivonen  
 Cornelius Wille  
 Loreto Zornoza  
 Horst Zuse
- University of Castilla-La Mancha, Ciudad Real, Spain  
 FUNDP, Namur, Belgium  
 University of Idaho, Moscow (Idaho), USA  
 GSO-Hochschule, Nuremberg, Germany  
 ENSIAS, Morocco  
 Florida Atlantic University, USA  
 Technical University Cottbus, Cottbus, Germany  
 Alcatel-Lucent, Nuremberg, Germany  
 Fraunhofer IESE, Kaiserslautern, Germany  
 Robert Bosch GmbH, Stuttgart, Germany  
 DPO, Rome, Italy  
 Schueco-Service GmbH, Bielefeld, Germany  
 University of Alcalá, Spain  
 Fraunhofer IESE, Kaiserslautern, Germany  
 Concordia University, Montréal (Québec), Canada  
 Technical University of Valencia, Spain  
 Siemens AG, Munich, Germany  
 Consultant, Rome, Italy  
 Berlin School of Economics, Germany  
 University of Sfax, Tunisia  
 SES, Munich/Budapest, Germany/Hungary  
 Software Measurement Service Ltd., Edenbridge, UK  
 Zayed University, Abu Dhabi, UAE  
 Nokia Siemens Networks, Finland  
 University of Applied Sciences, Bingen, Germany  
 IBM, Spain  
 Technical University Berlin, Berlin, Germany

Sponsors



**Rexroth**  
**Bosch Group**

Organizers



**École de technologie supérieure**



## Table of Contents

### IWSM / Mensura 2009 Proceedings

Ensuring Reliability of Information Provided by Measurement Systems <i>Miroslaw Siarow and Wilhelm Meiding</i>	1
Analysis of the Functional Size Measurement Methods Usage by Polish Business Software Systems Providers <i>Beata Czarnacka-Chrobot</i>	17
Leveraging People-Related Maturity Issues for Achieving Higher Maturity and Capability Levels <i>Luigi Buglione</i>	35
A General Model for Measurement Improvement <i>Reiner Durnke, Hashem Yazbek, Evan Asfoura, and Konstantina Georgieva</i>	48
What Are the Significant Cost Drivers for GOSMIC Functional Size Based Effort Estimation? <i>Sohail Shahid Bajwa and Cigdem Gencel</i>	62
Analysis of the Risk Assessment Methods - A Survey <i>Konstantina Georgieva, Ayaz Farooq, and Reiner R. Durnke</i>	76
Cockpit Based Management Architectures <i>Robert Neumann, Fritz Zbrog, and Reiner R. Durnke</i>	87
A Case Study in COSMIC Functional Size Measurement: The Rice Cooker Revisited <i>Luigi Lavazza and Vieri Del Bianco</i>	101
Functional Size of a Real-Time System <i>Jean-Marc Desharnais, Alain Abran, Petar Efe Distic, Mert Can Ihs, and Frjan Nuri Karaca</i>	122
A Prototypical Simulation Model to Analyze the Business Process Performance <i>Andreas Schmietendorf and Andreas End</i>	130
Improvement Opportunities and Suggestions for Benchmarking <i>Cigdem Gencel, Luigi Buglione, and Alain Abran</i>	144

Functional Size Measurement Quality Challenges for Inexperienced Measurers .....	157
<i>Sylvie Trudel and Alain Abran</i>	
Applying Test Case Metrics in a Tool Supported Iterative Architecture and Code Improvement Process .....	170
<i>Matthias Vanden, Horst Lichter, and Tobias Röttschke</i>	
Towards an Early Software Effort Estimation Based on Functional and Non-Functional Requirements .....	182
<i>Mohamed Kussab, Maya Danewa, and Olga Ormandjieva</i>	
Formalization Studies in Functional Size Measurement: How Do They Help? .....	197
<i>Baris Ozkan and Onur Demirors</i>	
Using Models to Develop Measurement Systems: A Method and Its Industrial Use .....	212
<i>Miroslaw Staron and Wilhelma Meding</i>	
Evaluating Process Quality Based on Change Request Data - An Empirical Study of the Eclipse Project .....	227
<i>Holger Schackmann, Henning Schaefer, and Horst Lichter</i>	
Empirical Evaluation of Hunk Metrics as Bug Predictors .....	242
<i>Javed Ferzund, Syed Nadeem Ahsan, and Franz Wotawa</i>	
Using Support Vector Regression for Web Development Effort Estimation .....	255
<i>Anna Corazza, Sergio Di Martino, Filomena Ferrucci, Carmine Gravino, and Emília Mendes</i>	
A Comparison of Neural Network Model and Regression Model Approaches Based on Sub-functional Components .....	272
<i>Seçkin Tunaliilar and Onur Demirors</i>	
Formal Definition of Measures for BPMN Models .....	285
<i>Luis Reynoso, Eivara Rolón, Marcela Genero, Félix García, Francisco Ruiz, and Mario Piatini</i>	
Using Tabu Search to Estimate Software Development Effort .....	307
<i>Filomena Ferrucci, Carmine Gravino, Rocco Oliveto, and Federica Sarro</i>	
An Experimental Study on the Reliability of COSMIC Measurement Results .....	321
<i>Erdur Ungar, Onur Demirors, Özden Özcan Top, and Barış Özkan</i>	

Assessing the Documentation Development Effort in Software Projects .....	337
<i>Isaac Sánchez-Rosado, Pablo Rodríguez-Soria, Borja Martín-Herrera, Juan José Cuadrado-Gallego, and Javier Martínez-Herráz, and Alfonso González</i>	
Author Index .....	347

# Formal Definition of Measures for BPMN Models

Luis Reynoso<sup>1</sup>, Elvira Rolón<sup>2</sup>, Marcela Genero<sup>3</sup>, Félix García<sup>3</sup>, Francisco Ruiz<sup>2</sup>,  
and Mario Piattini<sup>3</sup>

<sup>1</sup> University of Comahue  
Buenos Aires 1400, Neuquén, Argentina

lreyoso@uncoma.edu.ar  
<sup>2</sup> Autonomous University of Tamaulipas  
Centro Universitario Tampico-Madero, 89336 Tampico, Tamaulipas, México  
erolon@inf-cr.uclm.es

<sup>3</sup> Department of Information Technologies and Systems  
Indra-UCLM Research and Development Institute, University of Castilla-La Mancha  
Paseo de la Universidad N° 4, 13071 Ciudad Real, Spain  
{Marcela.Genero,Felix.Garcia,Francisco.RuizG,  
Mario.Piattini}@uclm.es

**Abstract.** Business process models are currently attaining more relevance, and more attention is therefore being paid to their quality. This situation led us to define a set of measures for the understandability of BPMN models, which is shown in a previous work. We focus on understandability since a model must be well understood before any changes are made to it. These measures were originally informally defined in natural language. As is well known, natural language is ambiguous and may lead to misunderstandings and a misinterpretation of the concepts captured by a measure and the way in which the measure value is obtained. This has motivated us to provide the formal definition of the proposed measures using OCL (Object Constraint Language) upon the BPMN (Business Process Modeling Notation) metamodel presented in this paper. The main advantages and lessons learned (which were obtained both from the current work and from previous works carried out in relation to the formal definition of other measures) are also summarized.

**Keywords:** Business Process, BPMN, OCL, Measure, Formal Definition.

## 1 Introduction

In the last decade many organizations have found themselves being caught up in commercial environments of competitiveness and of constant change, both internally and externally. They therefore often have to update or modify their processes. This movement of organizations towards ongoing improvement is known as the BPR (Business Process Re-engineering) initiative, as proposed by Hammer and Champy in the nineties [1]. Nowadays, and thanks to the resource known as BPM (Business Process Management) which has been growing in popularity over the last few years, all the phases of the process life-cycle are being included, thus bringing together management theory and new technology [2].

The relevance of the business process is thus attaining more importance. This fact is evidenced by the appearance of several languages with which to model business processes. These languages are very different from each other, since each one studies the processes in a different way, depending upon the purpose for which it was created [3]. Among the existent languages, special attention must be paid to the following: IDEF 0 [4], IDEF 3 [5], UML 2.0 [6], and BPMN [7], as they are those which are most frequently used in industry.

Among the aforementioned languages, the BPMN (standard provides a notation which is widely understandable to all business users [7], and has thus caused this language to gain popularity. The BPMN standard is defined by the amalgamation of best practices within the business modeling community and standardizes a business process modeling notation and semantics of a Business Process Diagram (BPD). The definition of BPDs is due to the fact that business people are much more comfortable with visualizing business processes in a flow-chart format. BPMN follows the principle of readability of any tradition of flowcharting notation.

A BPD uses the graphical elements and those semantics that support these elements as defined in this specification [7]. The BPMN specification defines many semantic concepts used in defining processes, and associates them with graphical elements, markers, and connections. Due to the fact that the quantity of concepts is extensive, the graphical elements are divided into four basic categories of elements (see Fig. 1):

- Flow objects (the main graphical elements)
- Connecting objects (the means of connecting flow objects)
- Swimlanes (the means of grouping modeling elements)
- Artifacts (which provide additional information about processes)

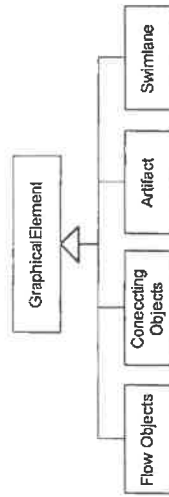


Fig. 1. Main BPMN Graphical Elements

These categories are further divided into sub categories. For example, there are two ways of grouping the primary 'swimlanes' modeling elements, by using Pools or Lanes (see Fig. 2). BPMN specifies all these concepts in detail, using class diagrams to describe the relation between the core BPMN graphical elements, their attributes, relationships and types.

The increasing relevance of BPMN models has caused several authors to focus on their quality [8, 9]. In a previous work we have defined a set of measures for the

understandability of BPMN models [10]. In order to obtain valid measures we followed a rigorous method defined in [11] and refined and extended in [12]. The method for measure definition currently being defined and refined within our research group is aligned with other existing proposals, such as that of [13]. These measures were initially defined in natural language (see Section 2). However, an informal definition in natural language may cause misinterpretations and misunderstanding, producing many undesirable effects, such as:

- Measures may not be repeatable: two different people applying the same measure to the same software artifact may attain two different results [14, 15].
- Experimental findings using the measure can be misunderstood due to the fact that it may not be clear what the measure really captures. Experiment replication is thus hampered [16].
- Measures extraction tools may attain different results [16].

Only a formal definition can avoid many of the aforementioned problems caused by imprecise definitions. One of the ways in which to achieve this is by means of the formal definition of measures using OCL upon a metamodel of the measured software artifacts [17]. Several works have been carried out in this area, e.g. in [17] a formal definition of OCL expression measures is presented, [18] deals with the formal definition of class diagram measures and [19] presents the formal definition of statechart diagram measures.

The formal definition provides the definition of measures in a formal language, and according to the Software Measurement Ontology [20], such definition corresponds to the "measurement approach".

The formal definition of a measure can be performed once (1) the measure is defined using natural language, and (2) both a metamodel and a formal language are selected (these activities are modeled in the UML activity model presented in Fig. 3). Furthermore, the formal definition of a measure should be coherent with its definition using natural language, i.e. the definition in natural language describes the way in which the value of a measure is obtained, and thus, its formal definition should not contradict that description.

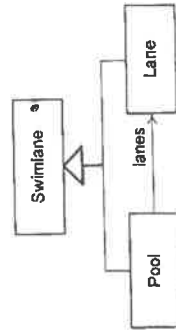


Fig. 2. Main BPMN Swimlane

The main goal of this paper is to formally define the measures for BPMN using OCL upon the BPMN metamodel elements presented in Appendix B of [21].

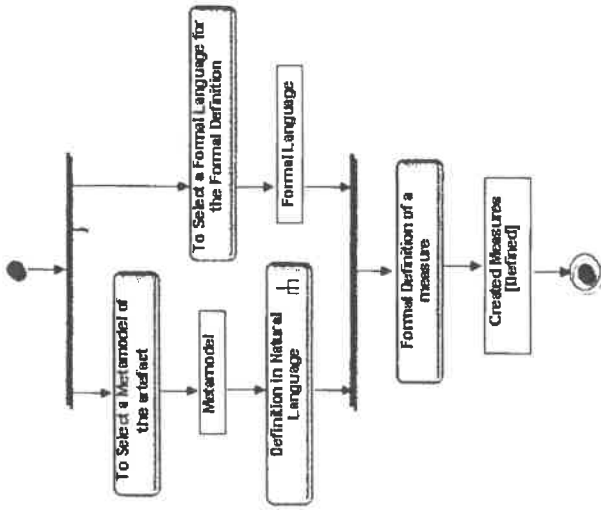


Fig. 3. Main activities of Measure Definition

The remainder of this paper is organized as follows: in Section 2 we present our previous work, Section 3 presents the formal definition of the measures for BPMN models, Section 4 includes some lessons learned, and finally, Section 5 outlines our main conclusions and future work.

## 2 Informal Definition

As we have mentioned, we are interested in measuring the understandability of BPMN models, but understandability is an external quality attribute that can only be measured when the models are finished. For this reason, indirect measures for understandability are needed, focusing on the structural properties of BPMN models, such as their structural complexity. Later empirical validation of these measures is needed to assess their capability to be used as early understandability indicators.

With the aim of measuring the structural complexity of BPMN models, we have proposed a set of measures in [10], which is divided into two categories: base measures and derived measures. The categories of base measures consist of 46 measures, which count the most significant elements of the BPMN metamodel. An example of the base measures related to gateways, connecting objects, swimlanes, artifacts and activities elements are shown in Tables 1 and 2 respectively. Measures related to events are

described in the Appendixes. Starting from these base measures, a set of 14 derived measures was defined which allowed us to see the proportions that existed between the different elements of the model. This set of derived measures is shown in Table 3. A more detailed description of the proposed measures is presented in [10].

Table 1. Base Measures for the Gateway/Connecting Objects, Swimlanes and Artifacts Elements

Core Element	Measure Name	Definition	Core Element	Measure Name	Definition
Exclusive Decision	NEIDDB	Number of Exclusive Decision/Data-based	Sequence Flow	NSF	Number of Sequence Flows in the Process
XOR Decision	NEDEB	Number of Exclusive Decision/merge Event-Based	Message Flow	NMF	Number of Message Flows between Participants in the Process
Inclusive (OR)	NID	Number of Inclusive Decision/merge	Pool	NP	Number of Pools in the Process
Complex	NCD	Number of Complex Decision/merge	Lanes	NL	Number of Lanes in the Process
Parallel (AND)	NPF	Number of Parallel Fork/join	Data Objects (Input)	NDOIn	Number of Data Object-In of the Process
			Data Objects (Output)	NDOOut	Number of Data Object-Out of the Process

Table 2. Base Measures for the Activity Element

Core Element	Measure Name	Definition
Task	NT	Number of Tasks
	NTL	Number of Task Looping
	NTMI	Number of Task Multiple Instances
	NTC	Number of Task Compensation
Collapsed Sub-Process	NCS	Number of Collapsed Sub-process
	NCSL	Number of Collapsed Sub-process Looping
	NCSMI	Number of Collapsed Sub-process Multiple Instance
	NCSA	Number of Collapsed Sub-process Ad-hoc

We shall now introduce an example with which to illustrate the calculation of the proposed measures. We will apply the measures to the BPMN model presented in Fig. 4. This model represents an engineering model for the design of a chip. The values obtained



### 3 Formal Definition

The formal definition of the measures is defined through derived attributes of the BPD. For example, in Fig. 5 we show two of the defined measures (NL and NP) (see Section 3.2.1) modeled through two derived attributes which have the same names as the measures. A query operation (*getGraphicalElements*) is defined in the Business Process Diagram metaclass which obtains the graphical elements contained in a BPD. This operation is defined using a definition constraint in Section 3.1.

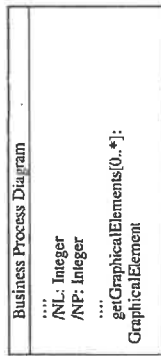


Fig. 5. BPD metaclass description for the measure definition

#### 3.1. A General Operation with Which to Obtain BPMN Graphical Elements

Fig. 6 shows a partial view of the main relationships between BPMN classes [21] which are used to understand how the *getGraphicalElement* operation of the Business Process Diagram metaclass is defined.

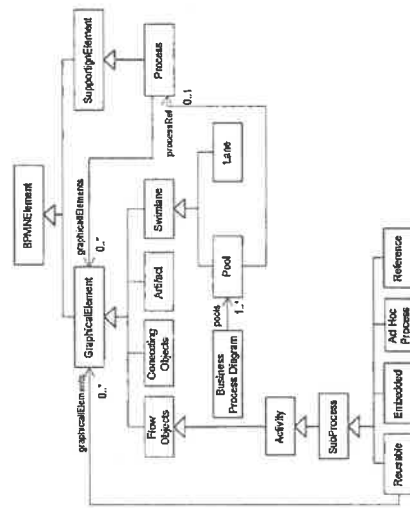


Fig. 6. A Simplified view of the main BPMN Graphical Elements and their relationships

A BPD contains one or more *Pools* (when there is only one pool in the diagram, its boundary may be invisible). Each *Pool* may have a *Process*. A business Process contains many graphical elements (e.g., *Events*, *Activities*, *Gateways*, and *Artifacts*). Thus, in order to obtain the set of graphical elements contained in a BPD the following operation can be defined:

```

context BusinessProcessDiagram
def: getGraphicalElement() : Set(GraphicalElement) = self.pools->
collect(p:pool | p->asSet()->
select (p | p.processRef.notEmpty()->
collect (p | p.processRef.graphicalElements)->flatten()

```

However, the previous operation does not take into account the fact that an activity may be an *Embedded sub-process* or an *Ad-Hoc Process*, which may contain a set of graphical elements (similar to composite objects). If we are to consider both situations, we thus need to define the *getGraphicalElement* in an appropriate manner:

- Firstly, we collect a set of graphical elements which are not *Embedded subprocess* (or *Ad-Hoc Process*), and
- Secondly, we add the graphical elements which are part of *Embedded subprocess* or *Ad-Hoc Process*. To obtain this, a recursion function should be defined, as *embedded subprocess* can also be defined in any *embedded subprocess*.

```

context BusinessProcessDiagram
def: getGraphicalElement() : Set(GraphicalElement) =
let elements: Set(GraphicalElement) = self.pools->
collect (p:pool | p->asSet()->
select (p | p.processRef.notEmpty()->
collect (p | p.processRef.graphicalElements)->
flatten()->
collect (g | not g.ocliisType(Embedded) or not g.ocliisType(Ad-Hoc
Process))->union elements->
collect (g | g.ocliisType(Embedded) or not g.ocliisType(Ad-Hoc
Process))->asSet()->
collect(x | x.getSubProcessElements()->
asSet()->context GraphicalElement
def: getSubProcessElements() : Set(GraphicalElement)
= if self.ocliisType(Embedded) then
self.graphicalElements->
collect (g | g.getSubProcessElements()->asSet()->
else
self
endif

```

#### 3.2 Definition of the Measures with OCL

The formal definition is presented according to two aspects:

- First, base measures are specified and then derived measures are specified.
- Base measures are presented according to the category each measure is related to. So, four subsections are defined: measures for *Swimlane*, measures for *Artifacts*, measures for *Flow Objects* and measures for *Connecting Objects*.

3.2.1 Base Measures for Swimlanes

This section describes the measures related to swimlanes graphical elements.

- Number of Lanes (NL). There must be one or more Lanes within a Pool [19] and a Pool includes 1 or more Lanes (see Fig. 7).

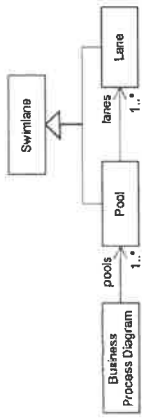


Fig. 7. Relationships between BPD, Pools and lanes [19]

```

context BusinessProcessDiagram
def: NL : Integer = self.pools.lanes-> count()
    
```

- Number of Participants (NP). According to [21], modelers must define the Participant for a Pool. The Participant can be either a Role or an Entity. A Pool has an association with a Participant Class, where an attribute of the Participant class identifies whether the participant is a role or an entity (see Fig. 8).



Fig. 8. Participants and Roles [21]

```

context BusinessProcessDiagram
def: NP: Integer = self.pools.participantref->
count(p | p.role->notempty())
    
```

3.2.2 Base Measures for Artifacts

- Number of Data Object-In of the Process (NDOIn). The Inputs attribute of the Process class defines the data requirements for input to the Process. Zero or more InputSets may be defined [21]. Each Inputset contains zero or more ArtifactInputs. An ArtifactInput is an Artifact, usually a Data Object (see Fig. 9).

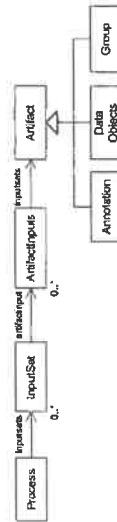


Fig. 9. Data Objects [19]

```

context BusinessProcessDiagram def: NDOIn : Integer =
self.getGraphicalElement()->
collect(e | e.oclistype(Process))->
count()
    
```

- Number of Data Object-Out of the Process (NDOOut). Similarly, the Outputs attribute of the Process class defines the data requirements for output to the Process.

```

context BusinessProcessDiagram def: pPOut : Integer =
self.getGraphicalElement()->
collect(e | e.oclistype(Process))->
collect(p.outputs)->
collect(a | a.artifactref.oclistype(Data*Object))->
count()
    
```

3.2.3 Base Measures for Connecting Objects

- Number of Sequence Flows in the Process (NSF). A Sequence Flow is a connection object (see Fig. 10).



Fig. 10. Main Connecting Objects [19]

```

context BusinessProcessDiagram
def: NSF : Integer = self.getGraphicalElement()->
count(e | e.oclistype(SequenceFlow))
    
```

- Number of Message Flows between participants in the process (NMF). A Message Flow is a connection object (see Fig. 8).

```

context BusinessProcessDiagram def: NMF : Integer =
self.getGraphicalElement()->
count(e | e.oclistype(MessageFlow))
    
```

3.2.4 Base Measures for Flow Objects

- Measures for Gateways. Gateways are modeling elements that are used to control how Sequence Flows interact as they converge and diverge within a process [21]. They are modeled through the hierarchy shown in Figure 11. The measures related to Gateways are formally specified in Table 6.

The value of *Start*, *Intermediate* and *Final Events* can be obtained through the following three specifications:

```

context BusinessProcessDiagram
def: TNSE() : Integer = self.getGraphicalElement-->
count(e | e.oclistType(Start) )
def: TNIE() : Integer = self.getGraphicalElement-->
count(e | e.oclistType(Intermediate) )
def: TNEE : Integer = self.getGraphicalElement-->
count(e | e.oclistType(End) )
    
```

A derived measure, TNE, is defined by using the previous specification

```

context BusinessProcessDiagram
def: TNE : Integer = TNSE + TNIE + TNEE;
    
```

However, it is possible to obtain the value of TNSE, TNIE, TNEE in terms of the base measures defined in Appendices A, B and C. The specification shown in these appendices uses two important attributes: *trigger* (an attribute which defines the type of trigger expected for a *Start/Intermediate Event*) and *result* (an attribute which defines the type of result expected for an End Event).

- Measures for Tasks. *Tasks* are modeled as a subclass of the *Activity* class (see Fig. 13). The *looptype* attribute is by default None, but may be set to Standard or MultiInstance [19]. The *isforcompensation* attribute is a boolean value to describe whether the activity is a compensate activity.

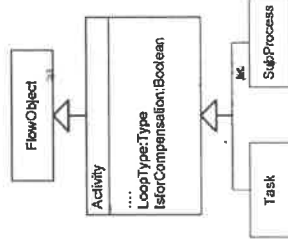


Fig. 13. Activities [19]

Table 7 shows the specification of the measures for Tasks.

Table 7. Measures for Tasks

Measure	Formal Definition
NT	<pre> context BusinessProcessDiagram def: NT : Integer = self.getGraphicalElement--&gt; collect(e   e.oclistType(Start) and e.LoopType = None)--&gt;count()                     </pre>

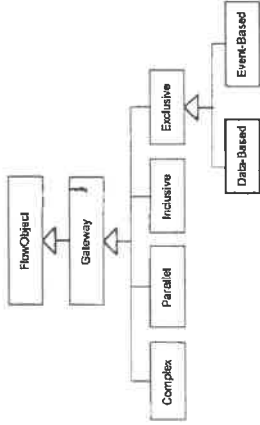


Fig. 11. Flow Objects [19]

Table 6. Measures for Gateways

Measure	Formal Definition
NEddb	<pre> context BusinessProcessDiagram def: NEddb : Integer = self.getGraphicalElement()-&gt; count(e   e.GatewayType = Exclusive and e.oclistType(Data-based) ) context BusinessProcessDiagram def: NEdeb : Integer = self.getGraphicalElement()-&gt; count(e   e.GatewayType = Exclusive and e.oclistType(Event-based) ) context BusinessProcessDiagram def: NID : Integer = self.getGraphicalElement()-&gt; count(e   e.oclistType(Inclusive) ) context BusinessProcessDiagram def: NCD : Integer = self.getGraphicalElement()-&gt; count(e   e.oclistType(Complex) ) context BusinessProcessDiagram def: NPF : Integer = self.getGraphicalElement()-&gt; count(e   e.oclistType(Parallel) )                     </pre>

- Measures for Events. The relationships between BPMN Event Elements are modeled in Fig. 12.

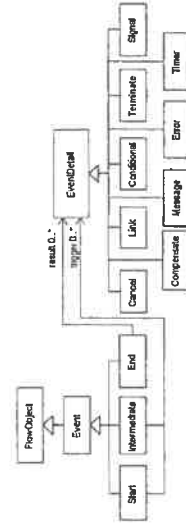


Fig. 12. Relationships between Events and EventDetails [19]

Table 7. (continued)

Measure	Formal Definition
NTL	<pre>context BusinessProcessDiagram def: NTL : Integer = self.getGraphicalElement-&gt; collect(e   e.occlisType(Task) and e.LoopyType = Standard)-&gt;count() context BusinessProcessDiagram def: NTMI : Integer = self.getGraphicalElement-&gt; collect(e   e.occlisType(Task) and e.LoopyType = Multiple)-&gt;count() context BusinessProcessDiagram def: NTC : Integer = self.getGraphicalElement-&gt; collect(e   e.occlisType(Task) and e.isforcompensation = true)-&gt;count()</pre>

- Measures for Collapsed Sub-Process. *SubProcesses* are a subclass of the *Activity* class (see Fig. 13). *LoopType* and *isforCompensation* attributes are used in the specification of the measures for *collapsed subprocesses* (Table 8). The *subprocesses* are modelled through a hierarchy of classes (see Fig. 14), in which the reusable classes are used to model the *collapsed subprocesses*.

Table 8. Measures for Collapsed Subprocesses

Measure	Formal Definition
NCS	<pre>context BusinessProcessDiagram def: NCS : Integer = self.getGraphicalElement-&gt; collect(e   e.occlisType(Sub-Process))-&gt; collect(s   s.subProcessType = reusable and s.loopyType = None)-&gt;count() context BusinessProcessDiagram def: NCSL : Integer = self.getGraphicalElement-&gt; collect(s   s.subProcessType = reusable and s.loopyType = standard)-&gt;count() context BusinessProcessDiagram def: NCSMI : Integer = self.getGraphicalElement-&gt; collect(e   e.occlisType(Sub-Process))-&gt; collect(s   s.subProcessType = reusable and s.loopyType = multiple)-&gt;count() context BusinessProcessDiagram def: NCSC : Integer = self.getGraphicalElement-&gt; collect(e   e.occlisType(Sub-Process))-&gt; collect(s   s.subProcessType = reusable and s.isforcompensation = true)-&gt;count() context BusinessProcessDiagram def: NCSA : Integer = self.getGraphicalElement-&gt;collect(e   e.occlisType(AdHocProcess))-&gt;count()</pre>

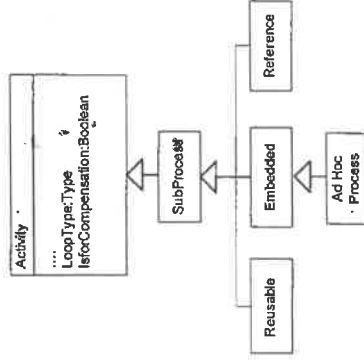


Fig. 14. SubProcess [19]

### 3.2.5 Derived Measures

Table 9 shows the specification of derived measures.

Table 9. Derived Measures

Measure	Formal Definition
TNSE	<pre>context BusinessProcessDiagram def: TNSE : Integer = self.NSNE + self.NSSTE + self.NSMSE + self.NSRE + ... + self.NSMSE context BusinessProcessDiagram def: TNIE : Integer = self.NINE + self.NITE + self.NIME + self.NIEE + self.NICoE + self.NICoE + self.NIRE + self.NILE + self.NIME context BusinessProcessDiagram def: TNBE : Integer = self.NEBE + self.NEMSe + self.NEBEE + self.NECE + self.NECoE + self.NELE + self.NEMUE + self.NETE context BusinessProcessDiagram def: TNT : Integer = self.NT + self.NTL + self.NTMI + self.NTC context BusinessProcessDiagram def: TNCS : Integer = self.NCS + self.NCSL + self.NCSMI + self.NCSC + self.NCSA context BusinessProcessDiagram def: TNE : Integer = self.TNSE + self.TNIE + self.TNEE context BusinessProcessDiagram def: TNG : Integer = self.NEDDB + self.NEDEB + self.NID + self.NCD + self.NPF context BusinessProcessDiagram def: TNDO : Integer = self.NDOIn + self.NDOOut context BusinessProcessDiagram def: CLA : Real = self.TNT.div(self.NSF)</pre>

Table 9. (continued)

Measure	Formal Definition
CLP	context BusinessProcessDiagram def: CLP: Real = self.NMF.div(self.NF)
PDOPIIn	context BusinessProcessDiagram def: PDOPIIn: Real = self.NDOIn.div(TNDO)
PDOPOut	context BusinessProcessDiagram def: PDOPOut: Real = self.NDOOut.div(self.TNDO)
PDOTOOut	context BusinessProcessDiagram def: PDOTOOut: Real = self.NDOOut.div(self.TNT)
PLT	context BusinessProcessDiagram def: PLT: Real = self.NL.div(self.TNT)

4 Lessons Learned

After carrying out the formal definition of the measures presented both in this paper and in other previous works [17, 19] we can provide the following suggestions:

1. The use of a software domain metamodel during the measure definition activity is a key aspect to consider. The definition of a measure has to be sufficiently clear and detailed, so that any concept of the software artifact (the object of study) mentioned in the natural language definition should be measurable. To fulfill this purpose a metamodel of the software artifact being measured should be selected as a previous activity of any measure definition. As is defined in [25], a metamodel constitutes the set of characteristics selected to represent a software or software piece and the set of their relationships, and these are proposed for the description of the software to which the measurement method will be applied. Consequently, the use of a metamodel will enable us: (1) to scrutinize whether any of the concepts mentioned in the measure definition (using natural language) was an element of the selected metamodel, and (2) to formally define each of the measures using a formal language.
2. We recommend OCL as a suitable language for the formal definition. OCL is becoming the *de facto* language with which to model constraints, it has been extensively used in modeling constraints for the UML language through its most recent versions (e.g. from UML 1.4 to 2.0), and it is used to define model transformations in the MDA (Model Driven Architecture) approach. Moreover, the formal definition of measures using OCL can be introduced in MDA compliant tools to extract the measures values for UML models.
3. Whenever possible, it is better to define generic operations for the formal definition of measures. In proposals which include measure definitions, it is usual to find that some of the measures are related to each other through shared concepts. In these situations it is useful to define generic operations which factorize operations

in order to facilitate the measure extractions by means of tools. For instance, in Section 3 we defined a general operation through which to obtain all the graphical elements within a BPD diagram. This operation, called *getgraphicalElements*, was reused in several measures definitions. It is important to factorize these generic operations by using the same concepts abstraction which is modeled in the metamodel (upon which the measures are defined). A similar approach towards defining generic operation was applied in the formal definition of OCL expressions measures [17] and statechart diagram measures [19].

4. With the formal definition of the measures for BPMN Models it was possible to identify the ambiguity in some measure definitions. One example of this is the NP measure. This base measure consists of counting the number of pools in a model, but by means of the formal definition of measures we can see that in a natural language the participant concept is not used, and when defining the measure formally it is possible to distinguish a participant and a role in a pool.

5 Conclusions

The main contribution of this paper is the formal definition of the measures for BPMN models proposed in [10] using OCL upon the BPMN metamodel built based on [21]. A formal definition of the measures is useful to obtain repeatable measures, i.e. measures which produce the same result each time when they are applied to a same artifact by a different person. The stakeholders within a business process thus benefit, along with all the people who use our BPMN measure as early indicators of BPD diagram understandability, who are provided with a precise definition of how the measure value is obtained, and no misunderstanding is introduced through its definition. Those people who build a measure extraction tool using the BPMN metamodel will also benefit as they can take advantage of transforming the formal definition of BPMN measures using OCL expressions to code by using MDA-compliant tools.

As part of our future work, we plan to align the current work using the latest version of the specification of BPMN 2 [26] which is currently being developed by OMG (Object Management Group).

Acknowledgements

This research is part of the MEDUSAS project (IDP-20090557) financed by the "Centro para el Desarrollo Tecnológico Industrial, Ministerio de Ciencia e Innovación" (CDTI), the PEGASOMAGO project (TIN2009-13718-C02-01) financed by "Ministerio de Ciencia e Innovación MICINN and Fondo Europeo de Desarrollo Regional FEDER", and the following projects financed by "Consejería de Ciencia y Tecnología de la Junta de Comunidades de Castilla-La Mancha": INGENIO (PAC 08-01-54-9262) and MECCA (PI2109-0075-8394).

## References

1. Hammer, M., Champy, J.: *Reengineering the Corporation: A Manifesto for Business Revolution*. Nicholas Brealey, London (1994)
2. Smith, H., Fingar, P.: *Business Process Management: The Third Wave*. Meghan-Kiffer Press, USA (2003)
3. Dufresne, T., Martin, J.: *Process Modeling for E-Business*. George Mason University (2003)
4. FIPS, *Integration Definition for Function Modeling (IDEF0)*. National Institute of Standards and Technology (1993)
5. Mayer, R.J., Menzel, C.P., Painter, M.K., de White, P.S., et al.: *Information Integration for Concurrent Engineering (IICE) IDEF3 Process Description Capture Method Report*. College Station, Texas (1995)
6. OMG, *Unified Modeling Language (UML) Specification: Infrastructure, version 2.0*. Object Management Group (2003)
7. OMG, *Business Process Modeling Notation (BPMN) Specification, Object Management Group (2006)*
8. Wohed, P., van der Aalst, W.M.P., Dumas, M., ter Hofstede, A.H.M., Russell, N.: *On the Suitability of BPMN for Business Process Modelling*. In: Duxidar, S., Fiadeiro, J.L., Sheth, A.P. (eds) *BPM 2006*. LNCS, vol. 4102, pp. 161–176. Springer, Heidelberg (2006)
9. Recker, J., Indulska, M., Rosemann, M., Green, P.: *Do Process Modelling Techniques Get Better? A Comparative Ontological Analysis of BPMN*. In: 16th Australasian Conference on Information Systems, Sydney, Australia, November 29 (2005)
10. Rolón, E., Ruiz, F., García, F., Piattini, M.: *Applying Software Metrics to evaluate Business Process Models*. *CLEI-Electronic Journal* 9(1), (Paper 5) (2006), <http://www.cleijournal.com/paper.php?id=117>
11. Calero, C., Piattini, M., Genero, M.: *Method for obtaining correct Metrics*. In: 3rd International Conference on Enterprise and Information Systems (ICEIS 2001), Setúbal, Portugal (2001)
12. Reynoso, L.: *A Measurement-Based Approach for Assessing the influence of Import-Coupling on OCL Expressions Maintainability*. Escuela Superior de Informática, Universidad de Castilla-La Mancha, Ciudad Real, Spain (2007)
13. Habra, N., Abran, A., Lopez, M., Sellami, A.: *A Framework for the Design and Verification of Software Measurement Methods*. *Journal of Systems and Software* 81(5), 633–648 (2008)
14. ISO/IEC, *9126 Software Product Evaluation—Quality Characteristics and Guidelines for their Use*. Geneva
15. Kitchenham, B., Pfleeger, S., Fenton, N.: *Towards a Framework for Software Measurement Validation*. *IEEE Trans. on Software Engineering* 21(12), 929–944 (1995)
16. Baroni, A.L.: *Formal Definition of Object-Oriented Design Metrics*. *Master of Science in Computer (2002)*
17. Reynoso, L., Cruz-Lemus, J.A., Genero, M., Piattini, M.: *OCL2: Using OCL in the Formal Definition of OCL Expression Measures*. In: 1st Workshop on Quality in Modeling QIM co-located with the ACM/IEEE 9th International Conference on Model Driven Engineering Languages and Systems (MODELS 2006), Genova, Italy (2006)
18. Baroni, A.L., Braz, S.: *Using OCL to Formalize Object-Oriented Design Metrics Definitions*. In: 6th International ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QUAOOSE 2002), Malaga, Spain (2002)
19. Reynoso, L., Cruz-Lemus, J.A., Genero, M., Piattini, M.: *Formal Definition of Measures for UML Statechart Diagrams Using OCL*. In: 23rd ACM Symposium on Applied Computing (SAC-SE 2008), Fortaleza, Ceará, Brazil, March 16–20. ACM, New York (2008)
20. García, F., Bertoa, M., Calero, C., Vallecillo, A., Ruiz, F., Piattini, M., Genero, M.: *Towards a consistent terminology for software measurement*. *Information and Software Technology* 48, 631–644 (2006)
21. OMG, *Business Process Modeling Notation (BPMN) Specification v 1.1 (draft)*. Object Management Group (2007)
22. Rolón, E., García, F., Ruiz, F., Piattini, M., Visaggio, C., Canfora, G.: *Evaluation of BPMN Models Quality: a Family of Experiments*. In: 3rd International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2008), Funchal, Madeira, May 4–7, pp. 56–63 (2008); 978-989-811128-9
23. <http://alarcos.inf-cr.uclm.es/bpmn/experiments/>
24. Rolón, E., Sánchez, L., García, F., Ruiz, F., Piattini, M., Caivano, D., Visaggio, G.: *Prediction Models for BPMN Usability and Maintainability*. In: IEEE Conference on Commerce and Enterprise Computing, Vienna, Austria, pp. 383–390 (2009)
25. Jacquet, J.P., Abran, A.: *From Software Metrics to Software Measurement Methods*. In: 3rd International Software Engineering Standards Symposium (ISESS 1997). IEEE Computer Society, Washington (1997)
26. OMG, *Business Process Model and Notation (BPMN) Specification 2.0, VO.9.7 (revised submission draft)* (March 6, 2009)

Appendix A

Base Measures for BPMN Start Event Elements

Measure	Definition	Formal Definition
NSNE	Number of Start None Events	<pre>context BusinessProcessDiagram def: NSNE: Integer = self.getGraphicalElement-&gt; count(e   e.oclisType(Start) and e.trigger-&gt; isEmpty())-&gt;count()  When the trigger attribute is empty (the EventDetail is not defined), this is considered a None End Event and the Event will not have an internal marker.</pre>
NSTE	Number of Start Timer Events	<pre>context BusinessProcessDiagram def: NSTE: Integer = self.getGraphicalElement-&gt; count(e   e.oclisType(Start) and e.trigger-&gt; isnotEmpty() and e.trigger.oclisType(Timer) )-&gt; count()</pre>
NSME	Number of Start Message Events	<pre>context BusinessProcessDiagram def: NSME: Integer = self.getGraphicalElement-&gt; count(e   e.oclisType(Start) and e.trigger-&gt; isnotEmpty() and e.trigger.oclisType(Message) )-&gt; count()</pre>
NSLE	Number of Start Link Events	<pre>context BusinessProcessDiagram def: NSLE: Integer = self.getGraphicalElement-&gt; count(e   e.oclisType(Start) and e.trigger-&gt; isnotEmpty() and e.trigger.oclisType(Link) )-&gt; count()</pre>
NSRE	Number of Start Rule Events	<pre>Context BusinessProcessDiagram def: NSRE: Integer = self.getGraphicalElement-&gt; count(e   e.oclisType(Start) and e.trigger-&gt; isnotEmpty() and e.trigger.oclisType(Rule) )-&gt; count()</pre>
NSME	Number of Start Multiple Events	<pre>Context BusinessProcessDiagram def: NSME: Integer = self.getGraphicalElement-&gt; count(e   e.oclisType(Start) and e.trigger-&gt; isnotEmpty() and e.trigger-&gt; size() &gt; 1 )-&gt;count()  If the trigger attribute contains more than one EventDetail, this is considered a Multiple End Event and the Event will have the star internal marker [BPMN]</pre>

Appendix B

Base Measures for BPMN Intermediate Event Elements

Measure	Definition	Formal Definition
NINE	Number of Intermediate None Events	<pre>context BusinessProcessDiagram def: NINE: Integer = self.getGraphicalElement-&gt; count(e   e.oclisType(Intermediate) and e.trigger-&gt;isEmpty())-&gt;count()</pre>
NITE	Number of Intermediate Timer Events	<pre>context BusinessProcessDiagram def: NITE: Integer = self.getGraphicalElement-&gt; count(e   e.oclisType(Intermediate) and e.trigger-&gt;isnotEmpty() and e.trigger.oclisType(Timer) )-&gt;count()</pre>
NIME	Number of Intermediate Message Events	<pre>context BusinessProcessDiagram def: NIME: Integer = self.getGraphicalElement-&gt; count(e   e.oclisType(Intermediate) and e.trigger-&gt;isnotEmpty() and e.trigger.oclisType(Message) )-&gt;count()</pre>
NIEE	Number of Intermediate Error Events	<pre>context BusinessProcessDiagram def: NIEE: Integer = self.getGraphicalElement-&gt; count(e   e.oclisType(Intermediate) and e.trigger-&gt;isnotEmpty() and e.trigger.oclisType(Error) )-&gt;count()</pre>
NICaE	Number of Intermediate Cancel Events	<pre>context BusinessProcessDiagram def: NICaE: Integer = self.getGraphicalElement-&gt; count(e   e.oclisType(Intermediate) and e.trigger-&gt;isnotEmpty() and e.trigger.oclisType(Cancel) )-&gt;count()</pre>
NICoE	Number of Intermediate Compensate Events	<pre>context BusinessProcessDiagram def: NICoE: Integer = self.getGraphicalElement-&gt; count(e   e.oclisType(Intermediate) and e.trigger-&gt;isnotEmpty() and e.trigger.oclisType(Compensate) )-&gt;count()</pre>
NIRE	Number of Intermediate Rule Events	<pre>context BusinessProcessDiagram def: NIRE: Integer = self.getGraphicalElement-&gt; count(e   e.oclisType(Intermediate) and e.trigger-&gt;isnotEmpty() and e.trigger.oclisType(Conditional) )-&gt;count()</pre>
NILE	Number of Intermediate Link Events	<pre>context BusinessProcessDiagram def: NILE: Integer = self.getGraphicalElement-&gt; count(e   e.oclisType(Intermediate) and e.trigger-&gt;isnotEmpty() and e.trigger.oclisType(Link) )-&gt;count()</pre>
NIMoE	Number of Intermediate Multiple Events	<pre>context BusinessProcessDiagram def: NIMoE: Integer = self.getGraphicalElement-&gt; count(e   e.oclisType(Intermediate) and e.trigger-&gt;isnotEmpty() and e.trigger-&gt;size() &gt; 1 )-&gt;count()</pre>

## Appendix C

Base Measures for BPMN Final Event Elements

Measure	Definition	Formal Definition
NENE	Number of End None Events	<pre>context BusinessProcessDiagram def: NENE: Integer = self.getGraphicalElement-&gt; count(e   e.oclistype(End) and e.result-&gt; isEmpty())-&gt;count()</pre>
NEMsE	Number of End Message Events	<pre>context BusinessProcessDiagram def: NEMsE: Integer = self.getGraphicalElement-&gt;count(e   e.oclistype(End) and e.result-&gt;isnotEmpty() and e.result.oclistype(Message) )-&gt;count()</pre>
NEEE	Number of End Error Events	<pre>context BusinessProcessDiagram def: NEEE: Integer = self.getGraphicalElement-&gt;count(e   e.oclistype(End) and e.result-&gt;isnotEmpty() and e.result.oclistype(Error) )-&gt;count()</pre>
NECaE	Number of End Cancel Events	<pre>context BusinessProcessDiagram def: NECaE: Integer = self.getGraphicalElement-&gt;count(e   e.oclistype(End) and e.result-&gt;isnotEmpty() and e.result.oclistype(Cancel) )-&gt;count()</pre>
NECoE	Number of End Compensation Events	<pre>context BusinessProcessDiagram def: NECoE: Integer = self.getGraphicalElement-&gt;count(e   e.oclistype(End) and e.result-&gt;isnotEmpty() and e.result.oclistype(Compensate) )-&gt;count()</pre>
NELE	Number of End Link Events	<pre>context BusinessProcessDiagram def: NELE: Integer = self.getGraphicalElement-&gt;count(e   e.oclistype(End) and e.result-&gt;isnotEmpty() and e.result.oclistype(Link) )-&gt;count()</pre>
NEMuE	Number of End Multiple Events	<pre>context BusinessProcessDiagram def: NEMuE: Integer = self.getGraphicalElement-&gt;count(e   e.oclistype(End) and e.result-&gt;isnotEmpty() and e.result-&gt;size() &gt; 1 )-&gt;count()</pre>
NETE	Number of End Terminate Events	<pre>context BusinessProcessDiagram def: NETE: Integer = self.getGraphicalElement-&gt;count(e   e.oclistype(End) and e.result-&gt;isnotEmpty() and e.result.oclistype(Terminate) )-&gt;count()</pre>