Politecnico di Milano

Polo di Como

Scuola di Ingegneria dell'Informazione

Corso di Laurea Specialistica in Ingegneria Informatica

BPMETRICS: a Software System for the Evaluation of Some Metrics for Business Process

Supervisor: Prof. Giuseppe POZZI

Master Graduation Thesis by:

Carlo CORTI Id. 729868

Academic Year 2010 - 2011

Abstract

It's during the 1990s that the workflow management prevailed as a new technology to support business processes and the term "process" became a new productivity paradigm but up to now many organizations have modeled and are modeling and designing business processes without the aid of metrics to question the quality of their models. As result, it may happen that simple processes are modeled in a complex and unsuitable way leading to a lower understandability, higher maintenance costs and inefficient execution of the process.

As modern organizations spend a lot of time creating and maintaining business processes, the importance of business process metrics is becoming increasingly important over time.

This thesis presents several business process metrics to evaluate the quality of a Business Process Model and all these metrics are available through the application BPMETRICS, in both web and standalone versions.

Keywords: business process, business process models, business process metrics, metrics, complexity, workflow metric, XPDL

Sommario

E 'nel corso degli anni 1990 che il workflow management si impose come una nuova tecnologia a supporto dei processi di business e il termine "processo" diventò un paradigma della nuova produttività, ma fino ad oggi molte organizzazioni hanno modellato e stanno modellando e progettando processi di business senza l'ausilio di metriche che definiscano la qualità dei loro modelli.
Come risultato, può accadere che i processi semplici sono modellati in modo complesso e inadatto portando ad una minore comprensibilità, maggiori costi di manutenzione e rendendo inefficiente l'esecuzione del processo.
Dal momento che le moderne organizzazioni spendono un sacco di tempo a creare e mantenere i processi di business, l'importanza delle metriche dei processi di business sta diventando sempre maggiore nel corso del tempo.
Questa tesi presenta diverse metriche di processi di business per valutare la qualità di un modello di business process e tutte queste metriche sono disponibili attraverso l'applicazione BPMETRICS, in entrambe le versioni stand-alone e web.

Parole chiave: processo di business, modelli dei processi di business, metriche dei processi di business, metriche, complessità, metriche dei workflow, XPDL

Table of Contents

List of Figures

Chapter 1

Introduction

This chapter provides a brief introduction to Business Process Management, the area in which the realization of this thesis arises, and Software Metrics, on which are based many of the Business Process Metrics. Will be finally presented the structure of this paper, briefly explaining the content of each chapter..

1.1 Business Software Management

In the last years, there has been a growing interest in business process management from industry as well as from business administration and systems research but the current management and improvement approach, with formal definitions and technical modelling, has been around since the early 1990s. In fact, it is only during the 1990s, that the workflow management prevailed as a new technology to support business process and the term “process” became a new productivity paradigm. The flourishing market and the availability of a wide range of products, in addition to allowing individual product vendors to focus on specific functional capabilities and users to take particular products to meet specific application needs, has revealed a fundamental problem: the lack of standards to enable the cooperation of different WFM products. To solve this problem, in 1993, a group of companies have joined to form the WFM Coalition that had as its primary purpose the establishment of standards and rules that allow the cooperation of different WFM products. In fact, the idea of creating standards and therefore developing appropriate specification for the implementation of workflow products arises from the fact that all the workflow

management systems have certain common characteristics with the purpose to enable the possibility to achieve the interoperability. In addition, these specifications, as well as allowing the interoperability between heterogeneous workflow products, has also improved the integration of workflow applications with other IT services (electronic mail, document management ...) but overall has improved the opportunities for the effective use of workflow technology within the IT market.

Since the 1990s the BPM market, and in particular the market BPM-related software, has always been expanding to become today a considerable market for software vendors, IT service providers and business consultants. In fact, market research have revealed that organizations that have had the best results in implementing business process management have spent more than 40% of the total time of the project in the study and implementation of the initial process model. In fact, the Business Process Modeling was considered one of the top 10 technology strategies in 2008 [1].

Business Process Modeling aims at producing a model of one or more business processes by defining the ways in which operations are carried out to achieve the objectives of an organization, known as Business Process Model. It may be constructed in multiple layers as the business process may be composed not only by a single process but by multiple nested processes. This model is an abstraction of reality and is used to describe the workflow or the integration between business processes. In particular, when we talk about Workflow we refer to the automation of procedures where documents, information or tasks are passed between participants according to a defined set of rules to achieve an overall business goal. The majority of the workflows are normally organized, within the context of an IT system, in order to provide support for the procedural automation.

In essence, business process is a collection of related, structured activities or tasks that produce a specific service or product for a particular goal. Three types of business process exist:

- Management processes that govern the operation of a system
- Operational processes that constitute the core business and create the primary value stream (Marketing, Sales …)
- Supporting processes which support the core processes (Recruitment, Technical Support…)

From the real importance of the business processes and their representation within the area of workflow software and the objectives of standardization of the WfMC, it follows the birth of language XPDL in the late 90's. In fact the XML Process Definition Language (XPDL) is a standard designed to exchange the process definition, both the graphics and the semantics of a workflow business process. Due to this fact, with the XPDL standard is possible to interchange business process definitions between different workflow products [2]. Later on, in the 2004 WfMC endorsed the Business Process Model Notation, a graphical formalism to standardize the way that process definitions were visualized. Therefore, XPDL was extended specifically with the goal to be able to represent in XML all of the concepts present in a BPMN diagram.

1.2 Software Metrics

In the mid-1960's software engineers started to use metrics, a measure of some property of a piece of software or its specifications, to characterize the properties of their code. As [3] says it is typical to divide the attributes/metrics into internal and external ones. In particular, internal software attributes can be

measured based only on the knowledge of a software artifact, such as size, cohesion and complexity while external software attributes can only be measured if additional knowledge is available about the environment of the software and of the interaction between the software and the environment, such as maintainability.

The main purpose of software quality metrics is to support the design phase in order to obtain program designs that are better structured, both from the point of view of the complexity of the model and from the point of view of the readability of the model by the user. In addition, the software quality metrics may have numerous applications in activities such as cost estimation, software quality assurance testing and performance optimization.

Given the importance that software metrics cover, in recent years it has tried to find the corresponding metrics for Business Process. In fact, several researchers, like Cardoso, Vanderfeesten and Mendling, have already identified similarities between software programs and business process design and recognized the potential of quality metrics in business process Management [4].

In fact the metrics of business process were born as the adaptation of software metric to the definitions of Business Process and in particular, most of these metrics are adapted considering the scheme/graph of the Business Process. Subsequently these metrics have been improved or new ones have been designed and created specifically for the BP in order to provide more useful information for creating Business Process Model. These metrics, not only consider the pure graphical representation of BP, but as internal software metrics, they also take into consideration those elements, attributes and definitions that are based on the knowledge of the Business Process definition.

1.3 Thesis Outline

The thesis is structured as follows:

- Chapter 2 – Start of Art: The chapter presents the state of the art of the business process metrics by providing an historical perspective about their development and about the approaches to the business process metrics in recent years. It also presents a couple of projects / software that are strictly related to the metrics for business process models.
- Chapter 3 – Goals of the Thesis: The chapter discusses the goals of the thesis, the design constraints and the requirements gathering.
- Chapter 4 – Design Choices: The chapter analyzes the design choices made during the analysis of the problem from the point of view of the software and from the point of view of the metrics to implement. In particular it discusses software, libraries and frameworks used during the development of this project and explains every metric available within the software.
- Chapter 5 – Description of the System: The chapter describes the system developed and its architecture. The chapter also explains the data structures and illustrates two of the main algorithms used to compute some of the metrics.
- Chapter 6 - Results: The chapter presents the results of the software implemented, illustrating both the stand-alone application and the web application, providing also the main scenario for each of them. The chapter then describes the sample process used to test all the metrics developed, showing the results and the proof of their correctness.
- Chapter 7 – Conclusions and Future Work: The chapter discusses the conclusions and the future developments of the thesis.

Chapter 2

State of Art

This chapter presents the state of the art of the business process metrics by providing an historical perspective about their development. The chapter discusses the approaches to the business process metrics in recent years and in the end it also presents a couple of projects / software that are strictly related to the metrics for business process models.

2.1 Business Process Metrics

In the mid-1960's, as reported in [5], software engineers started to use metrics to characterize the properties of their code, providing a good analysis mechanism to assess the quality of the software program design. Although the research in this area progressed over the years, we must expect the beginning of the 90 for the first work done in the field of Business Process. In 1990, Lee and Yoon with [6, 7] have led a job on the definition of metrics for Petri net process models and their empirical validation. In their work the metrics proposed were divided into two groups: structural metrics, which included simple calculations of places, transitions and arcs of the control graph, and dynamic metrics that cover the number markings and the maximum and average number of tokens for the original and a reduced state space. It was however Nissen the first to introduce the measurement concepts for business process modelling and business process design [8, 9]. The metrics he proposed cover counts for distinct paths, hierarchy levels and nodes in the process model, cycles, diameter and parallelism as number of nodes divided by the diameter. After Nissen, three Indian researchers Tjarden, Narasmihan and Gupta operationalize four characteristics of a business process that need to be balanced: simplicity, flexibility, integration and

efficiency [10, 11] while Morasca proposed with [12] a set of metrics for Petri nets identifying size, length, structural complexity and coupling and for each of them he defined a set of axiomatic properties which a respective metric would have to fulfil. Later on, Latva-Koivisto in [13] proposed several complexity metrics for business process models including the Coefficient of Network Connectivity, Cyclomatic Number, Reduction Complexity Index but these metrics were not considered that much because the work from Latva-Koivisto was published only as a technical report.

Research in this field hasn't always followed the same direction for all. In fact over the years have been created different streams among which we must mention the stream of research that adapts coupling and cohesion concepts from software engineering to the business process modelling. In this stream, Reijers and Vanderfeesten [14] developed a set of coupling and cohesion metrics to guide the design of workflow processes while on another different stream, Cardoso centered his work around the adaptation of the cyclomatic number for business process to what he called Control Flow Complexity (CFC) [15, 16]. Differing from the Latva-Koivisto, Cardoso is calculating, with the CFC metric, the complexity of a model by summing up the split connectors weighted by the combination of output markings they can product. Another different direction was taken by the research conducted by a group including Canfora, Rolon, Garcia, Piattini, Ruiz and Visaggio extended the work related to the measurement of the software process. In particular, Canfora et al. in [17] presented a set of metrics and evaluate their suitability to serve as predictors for maintainability of the process model while on another side, Balasubramanian and Gupta, inspired by Nissen’s and Tjaden’s work, proposed a set of metrics to support business process design [18], In particular, this set includes metrics to quantify the degree of automatic decision making, role integration, activity parallelism and activity automation.

Not all of the aspect, during the last year, had the same relevance in this research field. In fact, one of those that was a bit left in the corner were the cognitive considerations that play an important role for understanding good design in software engineering. In fact, while Gruhn and Laue in [19, 20] adapt the cognitive weights from software engineering to business process models, Vanderfeesten et all define the cross-connectivity metric that aims to capture how difficult is to understand how two nodes in a process model relate to each other [21]. Another aspect that was left out is the modularity that builds insight on software design as it is related to the number, the size and the depth of nesting modules. Regarding this concept, the approach by Lassen and Van De Aalst identified structured components in arbitrary process models for the translation to structured BPEL and, even more important, these components can be used to describe the process model in a quantitative way.

What has been presented is the historical evolution of the research on metrics related to business Process model and the various aspects that have characterized the past 20 years. Research in this field is still in its infancy and over time arouses more interest. In fact, besides the metric, in recent years have been presented some approaches, that will be described below, in which metrics are collected and "catalogued" in accordance with certain characteristic.

2.2 Approaches

Now we will describe the approaches to Business Process metrics that over the last few years have been identified and adopted by many researchers.

2.2.1 Software-related Metric Approach

Cardoso et al. in [4] proposed an approach that focuses on software metrics. In fact, the software metrics, according to Conte, Dunsmore & Shen in [22] has as

main purpose to obtain program design that are better structured. The advantages that this brings in first place are that the overall program logic is easier to understand for both the programmers and the users and in second place are that the identification of the modules is easier, since different functions are performed by different modules, which makes the maintenance of the software program easier. According to Conte and Shepperd in [22, 23] the useful metrics for a better quality of design areas based on 5 principles:

- *Coupling* It measures the number of interconnections among the modules. The degree of coupling depends on how complicated the connections are and on the type of connections. It is hypothesized that programs with a high coupling will contain more error than programs with a lower coupling value.

- *Cohesion* It measures the relationships between elements within a module. It is hypothesized that programs with low cohesion will contain more errors than programs with higher cohesion.

- *Complexity* Measure relative to the number of control constructs and to the number of modules. With the increase of these values, increases the value of complexity. In fact it is hypothesized that the higher design complexity the more errors the design will contain.

- *Modularity* It represents the degree of modularization. It is hypothesized that a low modularity is generally relates to more errors than higher modularity.

- *Size* It represents the size of the software both from the point of view of the modules and of their nesting. In fact it has been hypothesized that a larger program will contain more errors of a smaller one.

Based on this approach, Cardoso et al. in [4] presents the Business Process metrics that have been developed, designed or adapted in recent years, using the same classification:

- *Coupling* It measures the number of interconnections among the modules of the model and it is highly related to degree and density metrics in network analysis [24]. An example is the average degree, also called Coefficient of Connectivity, described in [13], that refers to the average number of connections that a node has with other nodes of the process graph. In contrast, there is the density metric [Mendling], that relates the number of available connections to the number of maximum connections for the given number of nodes.

- *Cohesion* It measures the coherence within the parts of the model. The only cohesion metric, developed so far, is described in [25] and it looks at the coherence within the activities of the process model. In particular it focuses on the information processing in the process and takes a data oriented view. In fact, for each activity in the process model the total cohesion is calculated by multiplying the information cohesion and the relation cohesion of the activity and then the mean of all activity cohesion values gives the whole process cohesion value.

- *Complexity* It measures the ease and understandability of the design. This kind of metric can be used by business process analysts and process designers to analyze the complexity of the process and, if possible, develop simpler one.

- *Modularity* It measures the degree to which a design is split into several modules. As stated into [26] there are no business process metrics that measure the modularity of a business process design.

- *Size* It measures the size of the model. This kind of measure is considered very important to complement other forms of process analysis.

2.2.2 Axioms Approaches

Another approach is the one presented by Antonini et al in [3]. In fact, in this case, it is considered the similarity between software and business process but using a more rigorous method. Actually, Antonini considers only the internal qualities of software, omitting the outer ones, and in particular he considers the size, the structural complexity and the coupling as Business Process Measures. For each of them are defined some properties and in the case where a metric don't satisfies such property, this metric is not considered valid. Indeed Morasca in [27] defines these properties, called axioms. Since the process model is represented by graphs using the BPMN standard, the axioms are defined on a graph-based representation of the software artifact, called system. Considering Figure 2.1, the system S is formed by two modules M1 and M2 and E1, E2, E3 and E4 are the elements respectively of M1 and M2 connected via the relation R.

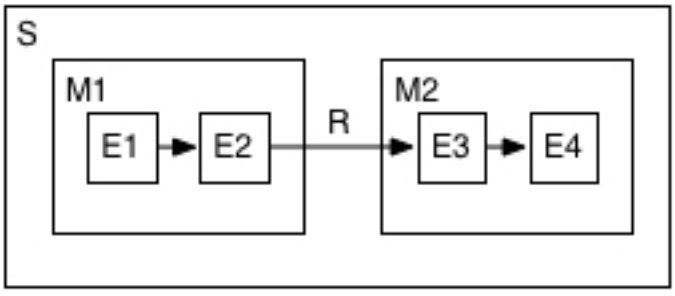


Figure 2.1 An example system

Based on the graph (Figure 2.1) the axioms are so defined:

Size:

- The overall size of S can never be greater than the sum of the sizes of M1 and of M2, if every element Ei belong either to M1, or to M2, or to both of them;
- The overall size of S equals the sum of the sizes of M1 and of M2, if every element Ei belongs either to M1, or to M2, but not to both of them.

Complexity:

- The overall complexity of S can never be smaller than the sum of the complexity of M1 and of M2, where S is composed by M1 and M2;
- The overall complexity of S equals the sum of the complexity of M1 and of M2, if M1 and M2 are two disjoint modules, with no connections from elements of one to elements of the other.

Coupling:

- The coupling of a module with no external relationship is null;
- If we add a new relationship R2 to an existing module M1, the coupling of M1 does not decrease;
- If we join the two modules M1 and M2, the coupling of the resulting module is never greater than the sum of the coupling of M1 and of M2;
- If we join two disjoint modules M1 and M2, the coupling of the resulting module is the sum of the coupling of M1 and of M2.

Thus, for every metric it should be checked that it conforms to the axioms and then is determined whether it is a measure of size, complexity or coupling. Once that is done, the measure is labeled according to its category. Antonini et al consider the metrics divided in 4 categories:

- Activity Attributes that consider the activities in a BP
- Control Flow Attribute that are defined based on the pure static structure of the graph of a BP
- Data flow attributes that address the flow of the information among the several activities involved in the graph of a BP
- Resource Attributes that consider the resources required and used by the graph of a BP during the process execution.

2.2.3 Category Approaches

The third and last approach, presented by Mendling in [5], is different from the others as it does not consider correlation between software metrics and metrics for BP, but considers them as exclusive metrics for Business Process and dividing them into six categories:

- *Size*, group of metrics related to the number of nodes of the process model;
- *Density*, group of metrics that relates the number of nodes to the number of arcs;
- *Partitionality,* group of metrics that refers to those aspect of a process

model that relate to the relationship of subcomponents to the overall model;

- *Connector Interplay*, group of metrics related to connectors (split and join and their 3 different type: and, or, xor) and their interplay;
- *Cyclicity*, group of metrics related to the cyclic parts of the model;
- *Concurrency*, group of metrics related to the concurrent paths of a business process model that need to be synchronized.

2.3 Relevant Projects

The applications that have been taken into account during this analysis are moving on two very different roads. The first one is presented by Mendling in [5] and takes into account the possible relations between the values given by the metrics and the presence of errors within a business process model. The second, however, is a tool that supports the design of business process models with the help of metrics.

2.3.1 Prediction of error probability based on metrics

Cardoso et al in [4] states that business process models which are designed using the business process metrics contain less error and are easier to understand and maintain. A first step made towards the validation of this hypothesis is made in a quantitative analysis about the connection between simple metrics and error probability in [5].

This survey proposed by Mendling is divided into two parts the verification of relaxed soundness, which is a minimal correctness criterion for a business process model, and the prediction error probability based on statistic methods.
The verification of relaxed soundness has brought to light that a part of the tested models (6% to about 600) contains errors and already this result suggests the need for verification tools for business process modelling projects of business process modelling projects.
In the second part, Mendling explain the possibility to use some of the business process metrics to predict the error probability. The author use a set of simple metrics related to size of the models as input to a logistic regression model. The results show that these metrics are suitable to predict the error probability and, in particular, it appears that a higher number of join-connectors is most strongly connected with an increase in error probability. But, as Cardoso reports in [4], there is a need for a further empirical investigation in order to establish an understanding of when a threshold value of a certain metrics indicates bad design in terms of maintainability or likely error proneness.

2.3.2 ProM tool

In last years, ProM has emerged as a powerful process analysis tool, supporting all kind of analysis related to business process. In contrast to many other tools, ProM uses process mining techniques and attempts to extract non-trivial and useful information from the so-called “event-logs” [4]. In fact, the process mining techniques use event logs as input in order to obtain the process model itself and, in particular, ProM offers many techniques to obtain the process model and the result may be a Petri net, EPC or a YAWL model. Otherwise, if the model is already given, the information stored in logs, can be used to check conformance, that is how well do reality and model fit together. This aspect can be seen as another quality dimension. Last but not least, ProM offers various

plug-ins to analyze the correctness of a model like soundness and absence of deadlocks and also allows the calculation of various quality metrics like cohesion, complexity, size, etc.

Chapter 3

Requirement Analysis

Considering the situation of the Business Process metrics and the software related to them, that were outlined in the previous chapter, in this chapter are explained the goals of the thesis, the project constraints and the few user requirements that are needed for a correct use of the BPMETRICS application and understanding correctly the metrics and their results.

3.1 Goals

The main goal of the thesis is to create a software application named BPMETRICS, that allows the assessment of Business Process Model in XPDL format, using quality metrics.

In order to achieve this goal, the work was divided into several parts:

- First of all, a literature search was performed of all available metrics for business process and part of these have been chosen only if they were "internal" metrics and considering their feasibility in terms of software development. Next, was performed an analysis phase in which were identified the characteristics of the system to be realized, both for the standalone version and for the web version.
- Next, we developed the BPMETRICS application, standalone version, that implements the metrics selected and which are described in Chapter 4.
- Subsequently was developed the BPMETRICS application, web version.

Finally, both the applications have been tested using some XPDL files created using the software TWE or publicly available on the website of the WfMC.

3.2 Project Constraints

The design constraints, that during the analysis phase of the problem have been imposed, concern for the most the technological part of the project. In fact, as design constraints, were imposed the use of Java as the programming language to build the BPMETRICS application standalone version and as consequence the use of the JSP technology to build the web version one.

As regards instead the Business Processes, have been required the use of the language XPDL as regards the definition of business process and consequently the BPMN notation for the creation of graphs. Now the constraints and their motivations will be described in detail.

3.2.1 Java

Java is the programming language Object Oriented used for the development of the application BPMETRICS.

Java was imposed as Project Constraints due to his strengths. In fact, as object-oriented programming language provides a high portability, allowing software to be made independent of the platform and the hardware on which it was made. Despite the high portability, maintains optimum performance and a good flexibility with regard to the graphics, desktop GUIs and web user interfaces. To remember, even the very good and well-thought-out exception handling and the huge library choice that Is available, like for example those aimed at the creation and manipulation of XML and PDF documents.

For the development of the BPMETRICS application was used Eclipse as working environment.

Finally, the last thing to consider is the fact that both Java and Eclipse are free and OpenSource.

3.2.2 XPDL

Xml Process Definition Language was imposed as Project Constraints for the Business Process part due to the fact that it's a standard file format for persisting BPMN diagrams and interchanging Process definitions and especially the BPMETRICS application supports the XPDL version 2.0 and XPDL version 2.1. In details, XPDL provides a standard graphical approach to Business Process Definition based on BPMN graphics. The file format is based on the WfMC meta-model which establishes a framework for defining, importing and exporting process definitions for numerous products including execution engines, simulators, BPA modelling tools, Business Activity Monitoring and reporting tools. One of the key elements of XPDL is its extensibility to handle information used by a variety of different tools. XPDL may never be capable of supporting all additional information requirements in all tools and that XPDL is a generic construct that supports vendor specific attributes for use within the common representation. Another characteristic is the possible mappings to specific execution languages (e.g. BPEL) and other XML-based specifications (e.g. ebXML). Finally, BPMN Model Portability conformance classes greatly increase the likelihood of true portability at the design level between a significant number of different vendor tools.

3.3 User Requirements

Below are listed the user requirements in order to use correctly the software and understand the results.

- The user must have knowledge, even basic, on the features and elements of the XPDL language, in order to understand certain metrics.
- The user must have knowledge regarding the Business Process Model Notation as it's necessary for the realization of business process which will then be used for the calculation of the metrics.
- Knowledge and meanings of the metrics proposed. For this purpose the application, both web and standalone version, offers a help section that allows the user to easy understand the meaning of each metric.

Chapter 4

Design Choices

In this chapter are discussed the design choices made during the realization of the BPMETRICS applications. All the choices are grouped into two categories: Software and Metrics. Both the groups are now explained in details.

4.1 Software

This groups relates to all those choices that are related to the technologies, libraries or frameworks used for the development of the application and to all the external software used to create test cases.

4.1.1 Together Workflow Editor

Together Workflow Editor (TWE) is the Workflow editor selected for the project as it fully implement the WfMC (Workflow Management Coalition) XPDL specifications. The TWE software was used to produce several Business Process Model that were used during the development phase but especially for the testing phase to check the correctness of the metrics developed.

Among all the various workflow editors, TWE was chosen for some of its characteristics:

- Pure Java application, usable on almost every operating system
- Transient XPDL package references
- View Referring elements for an XPDL elements
- On-line documentation including XPDL explanations and configuration
- Possibility to customize XPDL element's property panels

- View relation between main XPDL package and its external packages
- In-line property panels for editing XPDL
- Save XPDL with XPDL namespace prefix

4.1.2 Libraries and Framework

The framework and libraries used for the development of the BPMETRICS application will be now explained, defining also the reasons for which were chosen.

- *Struts2* is the framework used to develop the BPMETRICS application web version. In fact Struts 2 is an elegant, extensible framework for creating enterprise-ready Java web applications. This framework is designed to streamline the full development cycle, from building, to deploying, to maintaining applications over time.

- *iText* is the Java library used by the BPMETRICS application to generate the PDF reports of the metrics computed. This library was chosen for the BPMETRICS because it is used both in the standalone version and in the web version. It also allows the creation of dynamic PDF in an easy and intuitive way from a data structure in Java.

- *JAXB (Java Architecture for XML Binding)* is a Java libraries that allows developers to map Java classes to XML representations. JAXB provides two main features: the ability to *marshal* Java objects into XML and the inverse. In other words, JAXB allows storing and retrieving data in memory in any XML format, without the need to implement a specific set of XML loading and saving routines for the program's class structure.

JAXB is particularly useful when the specification is complex and changing. In such a case, regularly changing the XML Schema definitions to keep them synchronized with the Java definitions can be time consuming and error prone. This library is used inside the BPMETRICS to generate the XML reports of the metrics computed.

- *SAX (Simple API for XML)* is a programming interface, implemented in different languages, that allows to read and modify XML documents. Through SAX it's possible to implement specific XML parsers. SAX is used by the BPMETRICS application to load the XPDL file and convert it into the specific designed data structure (Chapter 5.2) and was preferred instead of DOM since SAX parsers have some benefits over DOM-style parsers:
    - SAX is event based, unlike DOM, and reacts to parsing events making report to the application.
    - A SAX parser only needs to report each parsing event as it happens, and normally discards almost all of that information once reported while a DOM parser builds a tree representation of the entire document.
    - The minimum memory required for a SAX parser is proportional to the maximum depth of the XML file and the maximum data involved in a single XML event while DOM parsers memory usage increases with the entire document length. This takes considerable time and space for large documents, like XPDL files.
    - SAX has better performance with large files due to its event-driven nature and processing documents is generally far faster than DOM-style parsers.

4.2 Metrics

The metrics that have been collected, selected and implemented are divided into 4 categories:

- *Activity Metrics*, that consider only the activities in a Business Process
- *Control Flow Metrics*, that are defined on the pure static structure of the graph of a Business Process.
- *Data Flow Metrics*, that consider the flow of information among the several activities involved in the graph of a Business Process.
- *Resource Metrics*, that consider the resources involved and used by the graph of a Business Process during the execution.

The metrics implemented and available within the application are the following:

- Activity Size
- Data Flow Size
- Data Flow Coupling
- Resources Size
- Resources Coupling
- Event Size
- Start Event Size
- End Event Size
- Intermediate Event Size
- Connector Size
- And Split Size
- And Join Size

- Or Split Size
- Or Join Size
- Xor Split Size
- Xor Join Size
- Control Flow Size
- Control Flow Complexity
- Diameter
- Density
- Coefficient of Connectivity
- Activity coupling
- Degree of Connectors
- Maximum Degree of Connector
- Separability
- Sequentiality
- Depth
- Average Depth
- Connector Mismatch
- Connector Heterogeneity
- Cycility
- Token Split

The metrics listed above will now be presented individually with a detailed description.

4.2.1 Activity Size

The Activity Size [3] metric represents the number of tasks of the Business Process. We must also consider that a BP may contain one (or more) supertask

or one (or more) subprocess and then we must consider the number of tasks contained by them. In fact, the size of a process is as follows:

$$Size_A(p) = \sum_{i=1}^{n_T} Size_A(task_i) + \sum_{i=1}^{n_{ST}} Size_A(supertask_i) \qquad (4.1)$$

where $Size_A(task_i) = 1$ and $Size_A(supertask_i)$ is the sum of the sizes of the n tasks in the supertask.

4.2.2 Data Flow Size

The data-flow size represent the amount of data (number of data items) managed by a process. As reported in [3], the data that a process could manage are divided in 4 kinds:

- Reference: these data (DR) univocally identify a process instance and the path followed by these data is the control flow of the graph of a Business Process.
- Operational: these data (DO) are needed by an activity for its processing and, in general, are not visible outside the task itself;
- Decision: these data (DD) are a subset of the operational data and are used by routing tasks to selectively activate the outgoing/incoming arc of the graph;
- Contextual: these data (DC) belong to a wider category of data, are relevant for the Business Process and typically include all the data managed by all the tasks of the process.

In particular the Data flow size of a process p is defined as:

$$Size_{DF}(p) = \sum_{j=1}^{n_T} V_{i,o}^{j} = \sum_{j=1}^{n_T} DR_{i,o}^{j} + \sum_{j=1}^{n_T} DO_{i,o}^{j} + \sum_{j=1}^{n_T} DC_{i,o}^{j} \qquad (4.2)$$

where n_T is the number of activities of the process and $V_{i,o}^{j}$ is the data recieved and produced by the node j.

4.2.3 Data Flow Complexity

The Data Flow Complexity take in consideration the data managed by a process but it treat this data in a different way that the Data Flow Size metric do. In fact this metric takes into consideration both a component deriving from the routing tasks and a component deriving from the tasks which set up a Business Process. As [3] states, this distinction is made due to the fact that routing task (RT) have a lower complexity if compared with a normal task (T) with the resulting overall complexity for data flow defined as the sum of the 2 components:

$$Complexity_{DF}(p) = \sum_{j=1}^{n_{RT}} Complexity_{DF}(RT_j) + \sum_{j=1}^{n_T} Complexity_{DF}(T_j) \qquad (4.3)$$

where

$$Complexity_{DF}(RT_j) = 1$$
$$Complexity_{DF}(T_j) = V_{i,o}^{j} \qquad (4.4)$$

4.2.4 Resources Size

The Resource Size [3] take into consideration all the resources required and used by the graph of a Business Process during process execution and is defined as follow:

$$Size_R(p) = \sum_{i=1}^{n_T} r_i = R \tag{4.5}$$

where n_T is the number of activities of the process and R are the resources available for the excution of the activities.

4.2.5 Resources Coupling

The Resource Coupling [3] is a metric that put in relation the "resources" and the relation between the activities within them. In fact, it considers the number of arcs which cross two (or more) swim lanes: every crossing means that the work item requires a new and different resources for its execution.

$$Coupling_R(p) = H \tag{4.6}$$

where H is the number of the transition between two activities that belong to a different swim lane.

4.2.6 Event Size

The Event Size [5] is a metric that counts the number of the Events in the Business Process.

$$Size_E(p) = \sum_{t \in Event} Size_E(t) \tag{4.7}$$

where

$$Size_E(t) = 1 \tag{4.8}$$

4.2.7 Start Event Size

The Start Event Size [5] is a metric that counts the number of the Start Events in the Business Process.

$$Size_{SE}(p) = \sum_{t \in Event-start} Size_{SE}(t) \tag{4.9}$$

where

$$Size_{SE}(t) = 1 \tag{4.10}$$

4.2.8 End Event Size

The End Event Size [5] is a metric that counts the number of the End Events in the Business Process.

$$Size_{EE}(p) = \sum_{t \in Event-end} Size_{EE}(t) \tag{4.11}$$

where

$$Size_{EE}(t) = 1 \tag{4.12}$$

4.2.9 Intermediate Event Size

The Intermediate Event Size [5] is a metric that counts the number of the Intermediate Events in the Business Process. An Intermediate Event is an event that occurs after a Business Process has been started. It will affect the flow of the process, but will not start or directly terminate the process. Then, this type of event can be used to disrupt the normal flow through exception handling or show the extra work required for compensation. It's also used to show where messages that are expected or sent within the process or where delays are expected within the process.

$$Size_{IE}(p) = \sum_{t \in Event-intermediate} Size_{IE}(t) \tag{4.13}$$

where

$$Size_{IE}(t) = 1 \tag{4.14}$$

4.2.10 Connector Size

The Connector Size [5] metric counts the number of connectors, divided in And, Or and Xor.

$$\begin{aligned} ConnectorSize(p) \\ = \sum_{rt \in AND} Size_{AND}(rt) + \sum_{rt \in OR-split} Size_{OR}(rt) \\ + \sum_{rt \in XOR} Size_{XOR}(rt) \end{aligned} \tag{4.15}$$

where

$$Size_{AND}(rt) = 1$$
$$Size_{OR}(rt) = 1 \quad (4.17)$$
$$Size_{XOR}(rt) = 1$$

4.2.11 And Split Size

The And Split Size metric [5] counts the number of And Split connectors.

$$Size_{AS}(p) = \sum_{rt \in AND-split} Size_{AND-split}(rt) \quad (4.18)$$

where

$$Size_{AND-split}(rt) = 1 \quad (4.19)$$

4.2.12 And Join Size

The And Join Size metric [5] counts the number of And Join connectors.

$$Size_{AJ}(p) = \sum_{rt \in AND-join} Size_{AND-join}(rt) \quad (4.20)$$

where

$$Size_{AND-join}(rt) = 1 \quad (4.21)$$

4.2.13 Or Split Size

The Or Split Size metric [5] counts the number of Or Split connectors.

$$Size_{OS}(p) = \sum_{rt \in OR-split} Size_{OR-split}(rt) \tag{4.22}$$

where

$$Size_{OR-split}(rt) = 1 \tag{4.23}$$

4.2.14 Or Join Size

The Or Join Size metric [5] counts the number of Or Join connectors.

$$Size_{OJ}(p) = \sum_{rt \in OR-join} Size_{OR-join}(rt) \tag{4.24}$$

where

$$Size_{OR-join}(rt) = 1 \tag{4.25}$$

4.2.15 Xor Split Size

The Xor Split Size metric [5] counts the number of Xor Split connectors.

$$Size_{XS}(p) = \sum_{rt \in XOR-split} Size_{XOR-split}(rt) \tag{4.26}$$

where

$$Size_{XOR-split}(rt) = 1 \tag{4.27}$$

4.2.16 Xor Join Size

The Xor Join Size metric [5] counts the number of Xor Join connectors.

$$Size_{XJ}(p) = \sum_{rt \in XOR-join} Size_{XOR-join}(rt) \tag{4.28}$$

where

$$Size_{XOR-join}(rt) = 1 \tag{4.29}$$

4.2.17 Control Flow Size

The Control Flow Size [3] of a process is defined as the number of activities and control elements of a process.

$$Size_{CF}(p) = \sum_{i=1}^{n_T} Size_{CF}(t_i) + \sum_{i=1}^{n_T} Size_{CF}(rt_i) \tag{4.30}$$

where

$$\begin{aligned} Size_{CF}(t_i) &= 1 \\ Size_{CF}(rt_i) &= 1 \end{aligned} \tag{4.31}$$

4.2.18 Control Flow Complexity

The Control Flow Complexity [3] for a process p is the sum of the complexities originated by the splits as:

$$\begin{aligned} Complexity_{CF}(p) \\ &= \sum_{rt \in AND-split} CFC_{AND-split}(rt) \\ &+ \sum_{rt \in OR-split} CFC_{OR-split}(rt) \\ &+ \sum_{rt \in XOR-split} CFC_{XOR-split}(rt) \end{aligned} \tag{4.32}$$

where

$$\begin{aligned} CFC_{AND-split}(rt) &= 1 \\ CFC_{OR-split}(rt) &= 2^{fan-out(rt)} - 1 \\ CFC_{XOR-split}(rt) &= fan-out(rt) \end{aligned} \tag{4.33}$$

fan-out is the number of task to which it is connected

4.2.19 Diameter

The Diameter [5, 28, 12] of a Business Process is the length of the longest path from a start node to an end node in the process model.

4.2.20 Density

The Density [5, 29] of a process graph is the number of arcs (A) divided by the number of the maximum number of arcs for the same number of nodes (N):

$$\Delta(G) = \frac{|A|}{|N| \cdot (|N| - 1)} \tag{4.34}$$

4.2.21 Coefficient of Connectivity

The Coefficient of Connectivity (CNC) [5, 13, 26] represents the ration of arcs (A) to nodes (N).

$$CNC(G) = \frac{|A|}{|N|} \tag{4.35}$$

4.2.22 Activity Coupling

The Activity Coupling (NCA) [5, 17] represents the ratio of nodes to arcs.

$$NCA(G) = \frac{|N|}{|A|} = \frac{1}{CNC} \tag{4.36}$$

4.2.23 Degree of Connectors

The Degree of a Connector [26, 30] is the number of nodes a connector is connected to. Two metrics relate to this concept:

- *Average Degree of Connectors* that represent the number of nodes a connector is in average connected to.

$$\overline{d_C}(G) = \frac{1}{C}\sum_{c \in C} d(c) \tag{4.37}$$

- *The Maximum degree of a connector* that is the highest degree of a connector inside a process.

$$\widehat{d_C}(G) = \max\{d(c) | \, c \in C\} \tag{4.38}$$

4.2.24 Separability

The Separability metric [5] is defined as the number of cut-vertices, that are those nodes in a graph whose deletion separates the process models into multiple and separated components, to the number of nodes.

$$\Pi(G) = \frac{|\{ n \in N \, | n \, is \, cut - vertex \}}{|N| - 2} \tag{4.39}$$

4.2.25 Sequentiality

The Sequentiality metric [5, 12] is the number of arcs between non-connectors nodes divided by the number of arcs.

$$\Xi(G) = \frac{|A \cap (T \times T)|}{|A|} \tag{4.40}$$

4.2.26 Depth

The Depth metric [5, 28] relates to the maximum nesting of structured blocks in a process. To calculate depth is define an algorithm that calculates the in-depth $\lambda_{in}(n)$ of a node n relative to its predecessor nodes n. All nodes are initialized with an in-depth value of 0 and at each node the updates the in-depth value $\lambda'_{in}(n)$ following these rules:

$$\lambda'_{in}(n) = \begin{cases} \max(\lambda_{in}(n), \lambda_{in}(pre) + 1)\ if\ pre \in S\ \wedge n\ \notin J \\ \max\big(\lambda_{in}(n), \lambda_{in}(pre)\big)\ if\ pre \in S\ \wedge n\ \in J \\ \max\big(\lambda_{in}(n), \lambda_{in}(pre)\big)\ if\ pre \notin S\ \wedge n\ \notin J \\ \max(\lambda_{in}(n), \lambda_{in}(pre) - 1)\ if\ pre \notin S\ \wedge n\ \in J \end{cases} \quad (4.41)$$

where

- S is the set of Splits
- J is the set of Joins
- $\lambda'_{in}(n)$ is the new in in-depth value
- $\lambda_{in}(pre)$ is the in-depth value of the previously visited predecessor node
- $\lambda_{in}(n)$ is the current in-depth value

There are two metrics related to this concept:

- Average Depth

$$\overline{\Lambda}(G) = \frac{1}{|N|} \sum_{i=1}^{N} \lambda_i\,(n) \quad (4.42)$$

- Maximum Depth

$$\Lambda(\mathrm{G}) = \max\{\lambda(n) \mid n\ \in N\ \} \quad (4.43)$$

4.2.27 Connector Mismatch

The Connector Mismatch metric (MM) [5] gives the sum of the mismatches for each connector type.

$$MM(G) = MM_{or} + MM_{and} + MM_{xor} \tag{4.44}$$

where

$$MM_l = | \sum_{c \in S_l} d(c) - \sum_{c \in J_l} d(c) | \tag{4.45}$$

4.2.28 Connector Heterogeneity

The Connector heterogeneity metric (CH) [5] refers to which extent different connectors are used in a business process model. For defining a suitable metric that ranges from 0 in the case that there are only connectors of one type, to 1 in the case that there are the same amount of connectors of all three types, we refer to the information entropy measure which has exactly these characteristics and defined as follows:

$$CH(G) = -\sum_{l \in \{and,or,xor\}} p(l) \cdot \log_3 p(l) \tag{4.46}$$

where

$$p(l) = \frac{|C_l|}{|C|} \tag{4.47}$$

4.2.29 Cyclicity

The Cyclicity metric [5] represent the ratio between the nodes on a cycle and the total number of nodes.

$$CYC_N = \frac{|N_C|}{|N|} \tag{4.48}$$

4.2.30 Token Split

The Token Split metric [5] sums up the output-degree $d_{out}(n)$ of AND-splits and OR-splits minues one.

$$TS(G) = \sum_{c \in C_{or} \cup C_{and}} d_{out}(n) - 1 \tag{4.49}$$

Chapter 5

Description of the System

Based on the specifications of the XPDL Workflow Management Coalition, has been implemented a Java application and a Web application, both named BPMETRICS, which are able to parse a XPDL file version 2.0 or 2.1 and evaluate those metrics that have been described in the chapter 4. However, this chapter describes the architecture of the application common to both versions, the data structure at the base of the application and two different algorithms that are used for the calculation of some metrics.

5.1 Architecture

The application that has been made, implements the metrics that have been described in Chapter 4 and in particular receives as input the XPDL file, created by the user with external software, and provides as output the metrics that are selected by the user.

This application has been made as a stand-alone application and as a Web Application.

Both applications made are divided into three main modules:

- Parsing in which has been implemented a parser dedicated to the XPDL files which converts the XPDL file contents in a data structure created specially to be able to compute the metrics;
- Metrics is the module in which the metrics have been implemented;
- Reports is the module in which have been implemented the two modes of reporting, as PDF file or as XML file, the results obtained by the application.

All three modules have been made in order to reduce the computational load and make the execution as light as possible to achieve the best performance of each of the modules. In particular way, in the Parsing module was used SAX and not DOM as the method for XPDL file parsing since SAX is way better then DOM for XPDL files (See Chapter 4.1.2), thus reducing the computational load of the first module. In the second module the metrics were implemented with algorithms that aim to optimize and reduce the computational costs since the metrics are calculated on a graph-based structure.

Communication via these modules is made through the data structure, designed ad hoc for this application, which is explained below.

5.2 Data Structure

Once the XPDL file is loaded and read by the SAX parser, it needs to be converted into a proper data structure in order to easy access the data contained from the source code. For this reason, an ad hoc data structure was created in order to store all the information that could be gathered from the XPDL file (Figure 5.1).

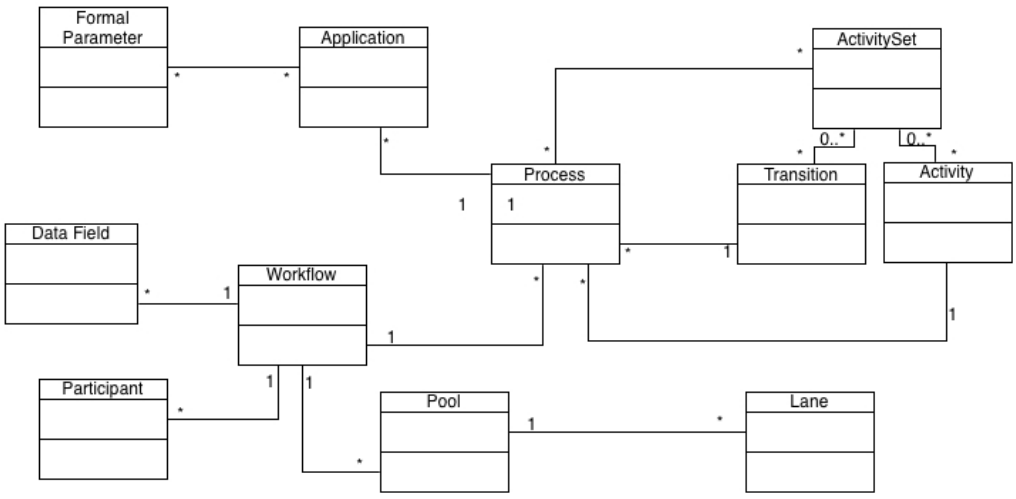


Figure 5.1 Diagram of the data structure ad hoc created for the BPMETRICS application.

The main entities of this structure are Process, Activity and Transition which respectively represent the generic process defined within a workflow, a generic activity contained within a process and a generic transition that binds two activity.
In particular, all these classes, except the Workflow, refer to a superclass called XPDLElement that defines the attributes common to all elements extracted from the XPDL file.
Below are described in detail the classes represented in the diagram.

Workflow This class represent the whole XPDL file. This class has as only attribute the xpdlVersion of the file. This class contains a list of Process, a list of Pool, a list of Participant and a list of Data Field.

XPDLElement This is an abstract class and represent the generic XPDL element except for the Workflow. The most significant attributes of this class are:

- *id*: is an attribute of type string that represents the unique id of the XPDL element to which it refers.
- *name*: this is an attribute of type string identifies the visual name of XPDL element.

Process This is a class that inherits from XPDLElement and defines a generic process defined in XPDL file read by the parser. This class contains a list of Activity and a list Transition.

Activity Even this class inherits from XPDLElement and represents the generic entity or node defined within a single process XPDL. This class contains lists of Extended Attribute and Parameter Actual that are used by the same Activity. The most significant attributes of this class are:

- *isRouting*: is a boolean attribute and identifies whether the node is a routing task or a simple task;
- *routingType*: is an attribute of type String and it identifies the type of routing element, that is if the routing task type is Exclusive (XOR), Parallel (AND) or Inclusive (OR)
- *routingPosition*: it is also an attribute of type String and it indicates the position of ruoting task in the Control Flow of the process, which indicates whether the routing task is a split or a join.
- *startEvent*, *endEvent*, *intermediateEvent*: These attributes are Boolean and respectively indicate whether the node/element in the process is a start event, an end event or an intermediate event.
- *subProcessId*: is an attribute of type String and indicates the sub-process id when the element is not a simple activity within the process but it represents a subprocess.

Transition This is a class that inherits from the generic XPDLElement and represents the transition between two entities or nodes within a process. The most significant attributes of this class are:

- *from*: this is an attribute of type String indicating the start activity of the transition.
- *to*: is an attribute of type String indicating the destination activity of the transition.

Participant This is a class that inherits from the generic XPDLElement and its definition represents an abstraction level between the real performer and the activity, which has to be performed. Only during run time this definition is evaluated and assigned to concrete humans and/or programs.

Pool This is a class that inherits from the generic XPDLElement and it represents a Participant in the process that can be a specific business entity or a more general business role. The Pool class contains the list of the lanes in which the it's divided and is characterized by these attributes:

- *mainPool*: this is a boolean attribute indicating if the pool is the main pool or not.
- *Process*: this is a String type attribute that indicates to which process the pool refers.

Lane Also this class inherits from the generic XPDLElement and it represent a sub-partition within a Pool. Lanes are often used for such things as internal roles and systems. The only attribute that the Lane class has is the performer attribute that indicates who is performing the activities within the lane.

DataField This is a class that inherits from the generic XPDLElement and it provides information about what the process does.

Application This is a class that inherits from the generic XPDLElement and it represents an application/service or tool required and invoked by the process defined within the process definition.

Formal Parameter This is a class that inherits from the generic XPDLElement and it represents a parameter that can be used as attribute in process or in application. These parameters are passed during invocation and return of control. The relevant attributes of this class are:

- *Mode*: is a String attribute and it indicates if the attribute is given as input or as output.
- *Type*: is a String attribute and it indicates the type of the attribute itself.

Extended Attribute This is a class that inherits from the generic XPDLElement and, as these attributes can be used in all entities (for this reason is not represented in the diagram), vendors use them to extend the functionality of the specification to meet individual needs. The only relevant attribute of this class is the *value* one that, as String attribute, indicates the value of the extended attribute defined in the XPDL file.

ActivitySet Also this class inherits from the XPDLElement class and it represent a self-contained set of activities and transitions. In particular the transitions in the set should refer only to activities in the same set and there should be no transitions into or out of the set.

5.3 Algorithms

In this section will be described two particular algorithms that are used for the calculation of the metrics. Respectively will be described the algorithm for the calculation of control flow paths within an oriented graph and the algorithm for the calculation of cut vertices inside an oriented graph.

5.3.1 Paths

The algorithm Path, implemented within the application, is used for the calculation of multiple metrics such as Diameter, Cyclicity and Depth. This algorithm allows to calculate all non-cyclic and cyclic paths, which has the beginning of a Start Event (node A in Figure 5.2) as a term and an End Event (nodes G and N in Figure 5.2), that exist within a directed graph, which in our case represent the XPDL process.

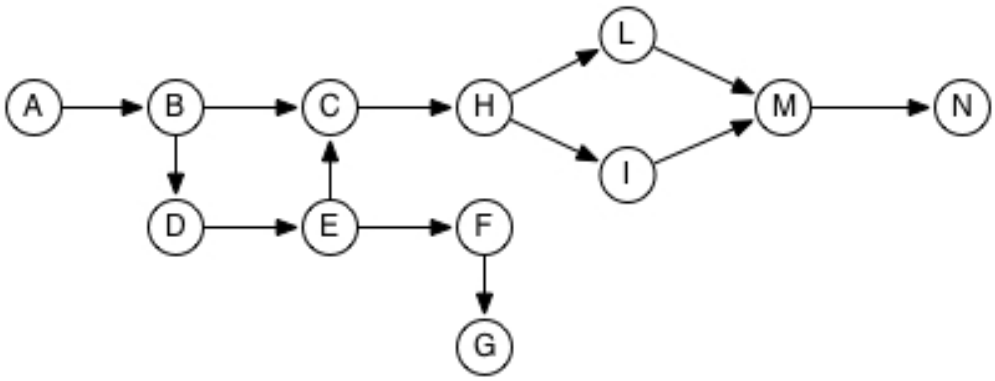


Figure 5.2 Graph based representation of a possible XPDL process

Below is reported the pseudo-code of the algorithm for the calculation of the path.

```
//Initialization of the variables
finalPaths[] = {};              // Path StartEvent-EndEvent
cyclicPaths[] = {};             // Cyclic Paths
currentPaths[] = {};            // Temporal Paths
startEvents = findStartEvents(process);
transitions = process.getTransitions();
//Conversion of the Start Events into Graph Paths
currentPaths = convertStartEventToPath(startEvents);
//Analyzing the paths till each of them end with an EndEvent or is cyclic
while(currentPaths.size() > 0) {
        path = currentPaths.get(0);
        lastVisitedActivity = path.getLastVisitedActivity();
        if(lastVisitedActivity.isSubProcess()) {
                // Computing the paths for the subprocess
                subProcessPaths = computePaths(subProcess);
                //Removing last element of the path as it stand for a sub-Process
                path.remove(lastVisitedActivity);
                //Creating a new path for each sub-Process path
                for(i=0; i < subProcessPaths.size(); i++) {
                        clonedPath =clonePath(path,subProcessPaths.get(i));
                        //Scanning transitions outgoing lastVisitedActivity
                        for(j=0; j < transitions.size();j++) {
                                transition = transitions.get(j);

                                if(transition.getFrom() == lastVisitedActivity) {
                                        // Destination of the transition
                                        transitionDestination = transition.getTo();
                                        // Cloning the path for each outgoing transition
                                        tempPath =  clonePath(clonedPath, transitionDestination);

                                        // If is an End Event the path is closed
                                        if(isEndEvent(transitionDestination, process)) {
                                                finalPaths.add(tempPath);
                                        // If is a cyclic one the path is discarded
```

```
                                        } else if(isCyclic(tempPath, transitionDestination)){

                                                cyclicPaths.add(tempPath);
                                        } else {
                                                currentPaths.add(tempPath);
                                        }
                                }
                        }
                }
        } else {
                //  Scanning each transitions outgoing lastVisitedActivity
                for(j=0; j < transitions.size();j++) {
                        transition = transitions.get(j);
                        if(transition.getFrom() == lastVisitedActivity) {
                                // Destination of the transition
                                transitionDestination = transition.getTo();
                                // Cloning the path for each outgoing transition
                                tempPath =  clonePath(clonedPath, transitionDestination);
                                // If is an End Event the path is closed
                                if(isEndEvent(transitionDestination, process)) {
                                        finalPaths.add(tempPath);
                                // If is a cyclic one the path is discarded
                                } else if(isCyclic(tempPath, transitionDestination)){
                                        cyclicPaths.add(tempPath);
                                } else {
                                        currentPaths.add(tempPath);
                                }
                        }
                }
        }
        //Removing the last analyzed path
        currentPaths.remove(0);
}
if(cyclicity) {
return cyclicPaths;
}
return finalPaths;
```

The algorithm begins with the collection of Event Start of the entire process, since they represent the first node of each possible path within the graph, and then inside *paths* we'll find as many paths as the Start Events. Then, for each cycle, the path (*currentPath*) that is currently topping the list *paths* is selected and the algorithm check if the last node of the *currentPath* is a sub-Process. If so, the algorithm compute the paths for the subprocess and add to the *currentPath* as many path as the sub-Process has. Then are taken into consideration all outgoing transitions from the last node within the currentPath. For each of them a path is created by duplicating the currentPath and adding to it

the destination node of the transition. Once you add the path to the destination node checks the type of this node and presented four scenarios:

- If the node is an End Even, the path is closed and is added to the list of *finalPaths*;
- If the node is a node already present inside the path, that path is recognized as recursive and is discarded, in the case where you do not seek recursive path;
- Otherwise it is added to the list paths.

Once completed the *currentPath* is removed from the *paths* list. The algorithm continues its execution until all currentPaths are visited and discarded because recursive or moved to *finalPaths*.

Considering the graph shown in Figure 5.2 and applying to it the algorithm above described, we obtain as a result of the following paths:

A → B → C → H → L → M → N
A → B → C → H → I → M → N
A → B → D → E → C → H → L → M → N
A → B → D → E → C → H → I → M → N
A → B → D → E → F → G

5.3.2 Cut Vertex

The algorithm CutVertex implemented within the application is used for the calculation of the Separability metric. This algorithm allows the identification of the cut vertices inside the graph representing the XPDL process. In particular, the cut vertices of a graph are those nodes that when are deleted from the graph, the graph is divided into two distinct graphs with no interconnections. Considering the graph shown in Figure 5.3, we can see how the cut vertices are the blue colored nodes.

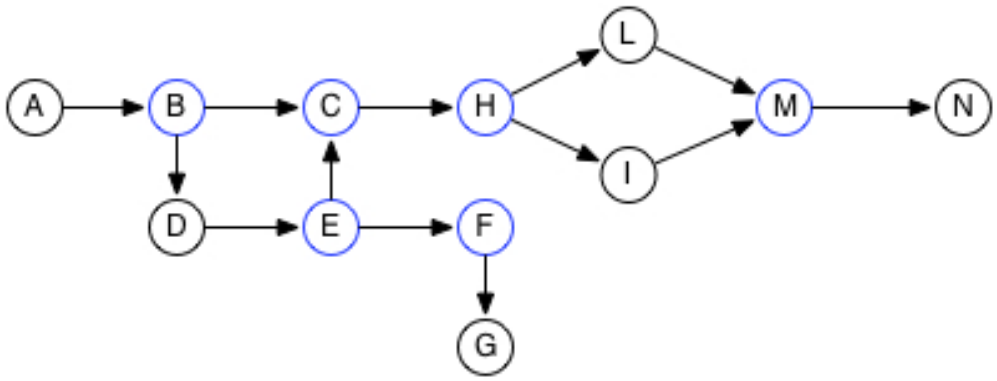


Figure 5.3 Graph based representation of a possible XPDL process, with the blue colored nodes representing the cut vertices of the graph.

Below is reported the pseudo-code of the algorithm for the calculation of the cut vertices.

```
counter = 0
low[] = {};
pre[] = {};
cutVertices[] = {};

//Initialization of variable
for (i=0; i < graph.getVertices(); i++) {
        low[i] = -1;
        pre[i] = -1;
}

for (i=0; i < graph.getVertices(); i++){
        if (pre[i] == -1) {
                dfs(graph, i, i);
        }
}

// Function that compute the Depth First Search algorithm
function dfs(Graph graph, int u, int v) {
   int children = 0;
   //Updating pre and low values of the v vertex
   pre[v] = counter++;
   low[v] = pre[v];
   //Analyzing the adjacent vertices of v
   for (int w : graph.adj(v)) {
      if (pre[w] == -1) {
         children++;
         dfs(graph, v, w);

         // update low number
         low[v] = Math.min(low[v], low[w]);

         // non-root of DFS is a cut vertex if low[w] >= pre[v]
```

```
            if (low[w] >= pre[v] && u != v)
                cutVertices[v] = true;
        } else if (w != u) {
                // update low number - ignore reverse of edge leading to v
            low[v] = Math.min(low[v], pre[w]);
        }
    }

    // root of DFS is a cut Vertex if it has more than 1 child
    if (u == v && children > 1)
        cutVertices[v] = true;
}
```

The algorithm begins with the definition and initialization of the *pre*, *low* and *cutVertices*. Then each node is visited and the *dfs* function is computed. This function gets as input the *graph* (that is already given as input to the algorithm) and the vertex *v* to analyze. This function updates the *low* and *pre* value of the *v* vertex and then check his adjacent vertices. For each of them, if they haven't been already processed, the *dfs* function is executed and then the *low* value is updated. Once the *low* value is updated it can be determine if the vertex *v* is or not a cut vertex by controlling these two conditions:

- a root vertex of DFS is a cut vertex if it has more than 1 child;
- non-root vertex of DFS is a cut vertex if the *low* value of its adjacent is higher than the node *pre* value.

Chapter 6

Results

This chapter discusses the results obtained by the development of the two applications. A scenario for both applications is proposed and a sample process is used as test bed. Obtained results are discussed.

6.1 Standalone Application

The BPMETRICS application consists of 4 main units: Menu, Options, File Loaded and Result.
The *Menu* is the means by which the user selects the operations, step by step , to perform:

- *File* Composed of the elements Open, Run and Quit allowing the user to, respectively, load and parse an XPDL file, run and calculate the metrics selected and close the application. The Run menu will be accessible to the user only after an XPDL file compliant to the project specification is selected and loaded.

- *Report* Made up of 2 elements: Generate PDF and Generate XML. These two elements allow respectively to generate a PDF or an XML report according to user needs and requirements. Both features will be available to the user only if the metrics, previously selected by the user, have been calculated and the results are displayed on the screen.

- *Metrics* This menu's unit provides access to the help section of the

application providing the user with a brief explanation for each of the metrics that are available within the application.

- *?* This menu's unit offers information about the author of the application in order to contact him.

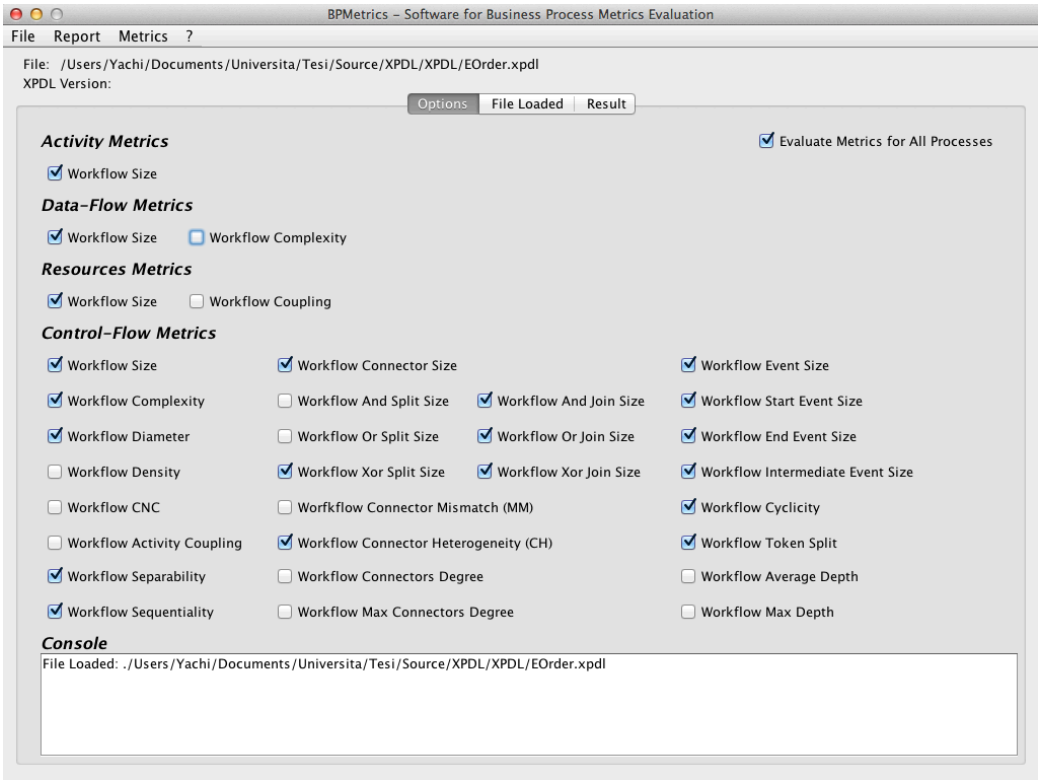
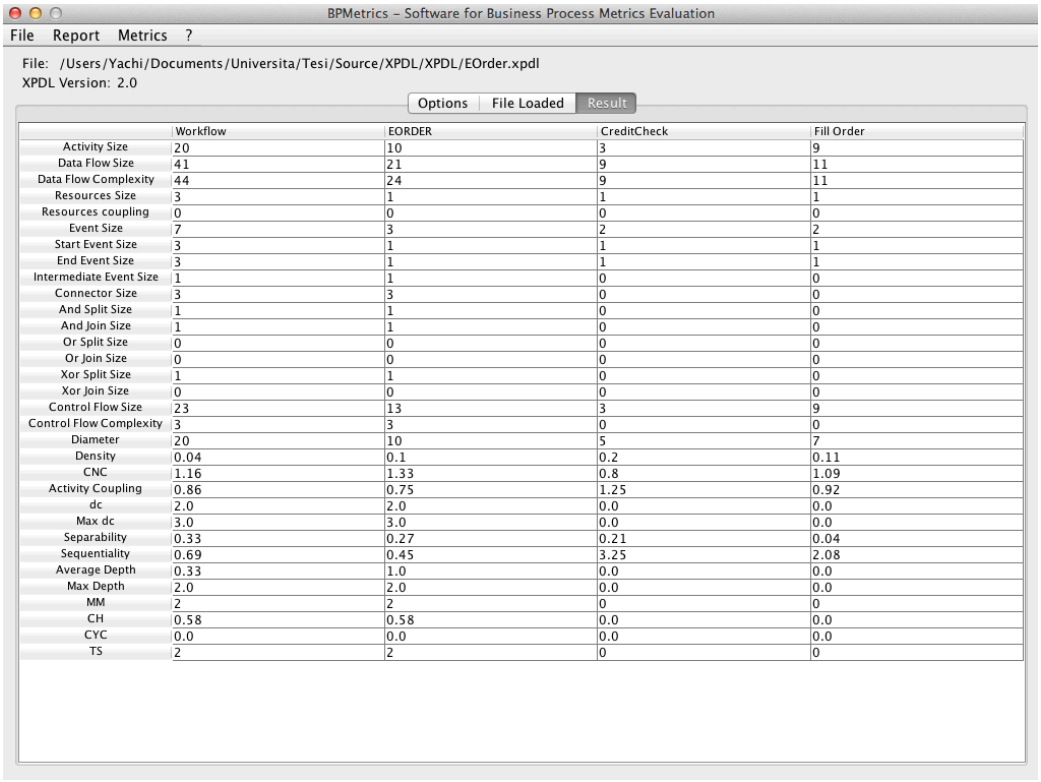


Figure 6.1 Option section of the BPMETRICS standalone application.

The Options section is the core of the application. In fact, as shown in Figure 6.1, this module allows the user to select the metrics desired and if these metrics need to be computed for each individual process within the XPDL file, previously loaded. There is also, always in this section, a small console that keeps track of the actions that are executed either manually by the user, or automatically by the application, to make the use of the application more transparent and clear.

The File Loaded section, instead, presents to the user the file XPDL loaded, in the case where there is the need to consult the file itself.

Results section, finally, presents the calculated metrics as result, as shown in the Figure 6.2.



	Workflow	EORDER	CreditCheck	Fill Order
Activity Size	20	10	3	9
Data Flow Size	41	21	9	11
Data Flow Complexity	44	24	9	11
Resources Size	3	1	1	1
Resources coupling	0	0	0	0
Event Size	7	3	2	2
Start Event Size	3	1	1	1
End Event Size	3	1	1	1
Intermediate Event Size	1	1	0	0
Connector Size	3	3	0	0
And Split Size	1	1	0	0
And Join Size	1	1	0	0
Or Split Size	0	0	0	0
Or Join Size	0	0	0	0
Xor Split Size	1	1	0	0
Xor Join Size	0	0	0	0
Control Flow Size	23	13	3	9
Control Flow Complexity	3	3	0	0
Diameter	20	10	5	7
Density	0.04	0.1	0.2	0.11
CNC	1.16	1.33	0.8	1.09
Activity Coupling	0.86	0.75	1.25	0.92
dc	2.0	2.0	0.0	0.0
Max dc	3.0	3.0	0.0	0.0
Separability	0.33	0.27	0.21	0.04
Sequentiality	0.69	0.45	3.25	2.08
Average Depth	0.33	1.0	0.0	0.0
Max Depth	2.0	2.0	0.0	0.0
MM	2	2	0	0
CH	0.58	0.58	0.0	0.0
CYC	0.0	0.0	0.0	0.0
TS	2	2	0	0

Figure 6.2 Result section of the BPMETRICS standalone application.

6.2 Web Application

The BPMETRICS application, which was developed on the Struts2 framework, consists of 4 units (pages): Home, Metrics, Metrics FAQ and Contact. The Home is just a brief presentation of the application outlining the its objectives and characteristics while the Contact unit allows the user to get in touch with the author of the application for any further details and / or explanations. Metrics

and Metrics FAQ instead represent the core part of the Web project. In fact, the unit Metrics is the unit that allows the user to upload his XPDL file and select the metrics to be calculated, as shown in Figure 6.3.

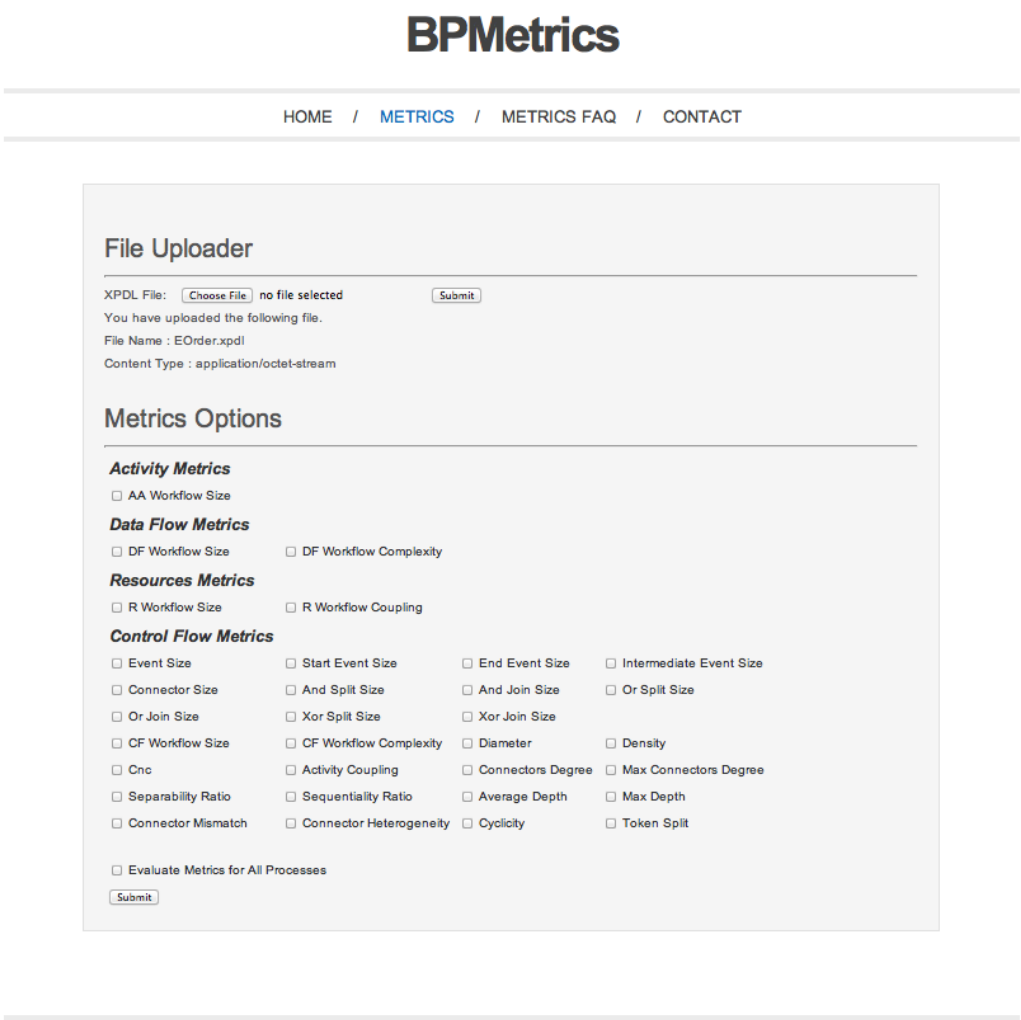


Figure 6.3 Metrics Unit of the BPMETRICS web application showing the panel to upload the XPDL file and the metrics selection panel

Once calculated the metrics, the results are displayed on screen in a tabular form and the user will be able to generate a PDF report or XML, depending on the needs or requirements (Fig 6.4).

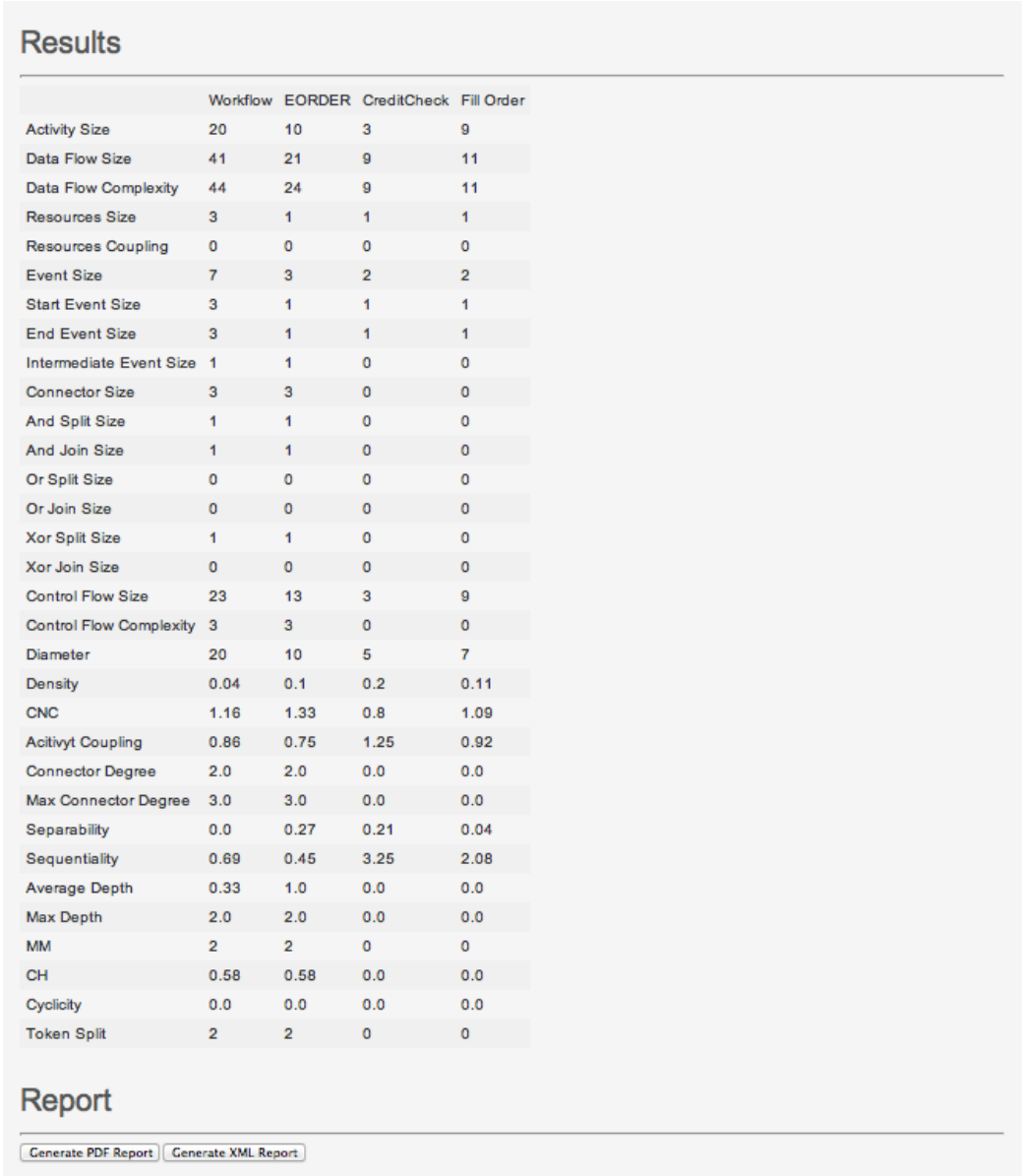


Figure 6.4 Metrics Unit of the BPMETRICS web application showing the results of the metrics in tabular form and the report section.

This Metrics unit has been created in order to follow the user step by step during its use. In fact, the contents are not all proposed at once but are available step by step. In fact, the user must first load the XPDL file and if that file will correspond to the project specifications (XPDL file version greater than 2.0), will be offered to the user the metrics panel in order to selected those required.

Finally the Metrics FAQ unit, as its corresponding module in stand-alone version, is a brief help explaining the meaning of the metrics developed that are available in the application to make easier the use of the application and to make clearer the comprehension of the metrics.

6.3 Scenarios

Both versions of the application, as already mentioned above, were made to follow the user, step by step, in its use. In fact, some features are not available when the application starts but are made available only successful execution of certain steps. For clarity will now be presented a complete scenario, for both for the standalone and we version, from the upload of the XPDL file to the final generation of reports.

6.3.1 Standalone scenario

- The user loads the XPDL file (Menu: File> Open)
- The XPDL file is parsed using SAX and it's created an instance of the Workflow class containing the elements of XPDL file. When finished, it will enable the possibility of calculating the metric (Menu: File> Run).
- The user selects the metrics (Panel Options) to perform and if he needed it, select the option to calculate the metrics for each process contained in the XPDL file.
- The user performs the calculation of the metric (Menu: File> Run) and creates an instance of the XPDLMetrics class that contains all the metrics calculated, that the user has previously selected.
- The calculated metrics are proposed in tabular form to the user (Result Panel) and it's enabled the possibility to generate reports.
- The user generates a PDF report (Menu: Report> Generate PDF) or

XML (Menu: Report> Generate XML).

During each phase, the user can refer to the help (Menu: Metrics) to better understand the meaning of each metric.

6.3.2 Web scenario

1. The user uploads the XPDL file
2. The UploadAction action, that loads the file, is invoked.
3. The XPDL file is parsed using SAX and it's created an instance of the Workflow class containing the elements of XPDL files.
4. Metrics.jps page appears confirming that the file is correctly loaded and parsed and it displays the metrics panel.
5. The user selects the metrics to be performed and in the case is needed, selects the option to calculate the metrics for each process.
6. The user performs the calculation of metrics
7. The XPDLMetricsAction action is invoked and calculates the selected metrics and creates an instance of the XPDLMetrics class than contains all the resulting metrics, that the user has previously selected.
8. The calculated metrics are proposed to the user in a tabular form and it's enabled the possibility to generate reports.
9. The user generates a PDF report or XML as needed.

During each phase the user can refer to the help (Metrics FAQ section) to better understand the meaning of each metric or contact the application author for further details or clarification.

6.4 Sample Process

For testing the application, we chose the EOrder sample process from the WfMC Specification. There are two version of the EOrder sample process: the 1.0 and the 2.0. As the application is made to be able to read only XPDL document past the version 2.0 we took in consideration only the Eorder sample process 2.0.

6.4.1 EOrder Process

The main process takes a formatted string as an input and returns a string that indicates whether the order was confirmed or rejected. It contains the following steps:

- The string is first converted to a complex data object. If an exception is caught (indicating that the string is incorrectly formatted), an alarm is raised and the order is rejected.
- The data is checked for accuracy.
- The process determines whether payment is via a purchase order or a credit card.
- Credit card orders are sent to a subprocess that authorizes the credit purchase.
- Purchase orders are validated by an application that checks the vendor‘s record and authorizes the purchase amount.
- The order is entered into the database and an order number is issued. The next three activities happen in parallel:
- An acceptance message is composed to return to the end user. A subprocess is invoked asynchronously to fill the order.
- An order confirmation email is sent to the end user. This activity is a

special activity that is managed by the system. It uses ExtendedAttributes to specify the information the system needs for the email.

- If an order is rejected, either because it is inaccurate or cannot obtain authorization, a rejection message is composed, to return to the customer.

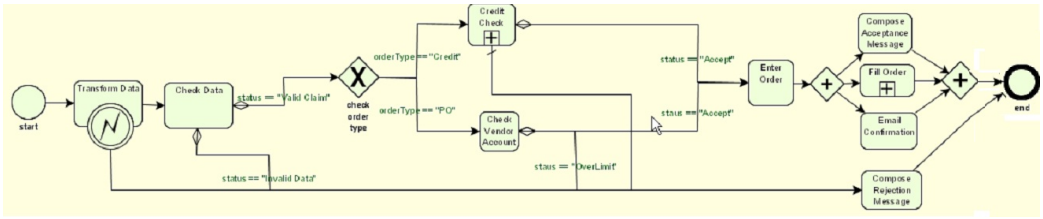


Figure 6.5 EOrder Main Process

6.4.2 Credit Check Sub-process

The CreditCheck subprocess sets up a CreditInfo object from the input parameters and then sends the information to a credit card web service for authorization. The web service returns a status string that is converted to an OrderStatus string and returned to the calling process.

Figure 6.6 The Credit Check sub-process

6.4.3 Fill Order Sub-process

This subprocess handles the shipping and billing of the order. This process includes a participant called a "Shipper"

- The first activity displays the order information to a Shipper who ships

the items in the order and records the status of the line items. The application returns the status of the order -- whether it is complete or backordered. This activity includes some deadlines. If the activity is not completed within 3 days, a notifyException is thrown an alarm is raised. If the activity is still not completed within 5 days, a timeoutException is thrown and the order is canceled.

- The process then determines if it is a PO or credit order. PO orders are sent to the billing system and then an electronic invoice is created and stored on a server.
- Credit card orders are sent to the credit card web service for charging and then an electronic receipt is created and stored on a server.
- The last step sends the invoice or receipt to the customer as an attachment to an email message. It uses ExtendedAttributes to specify the information needed for the email.

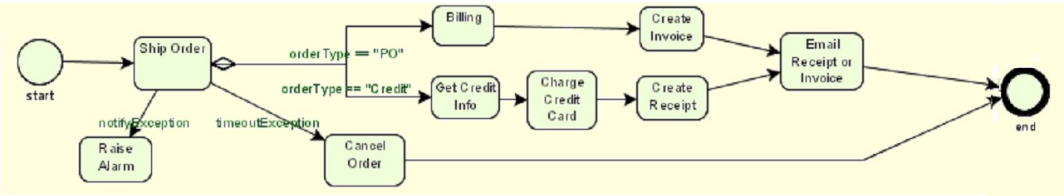


Figure 6.7 The Fill Order sub-process

6.5 Metric Results

The process described above was executed both with the stand-alone application and with the web application, obtaining the same results and therefore will be presented only the stand-alone results. The EOrder.xpdl file has been loaded and parsed from BPMETRICS and metrics have been selected to perform as shown in Figure 6.8.

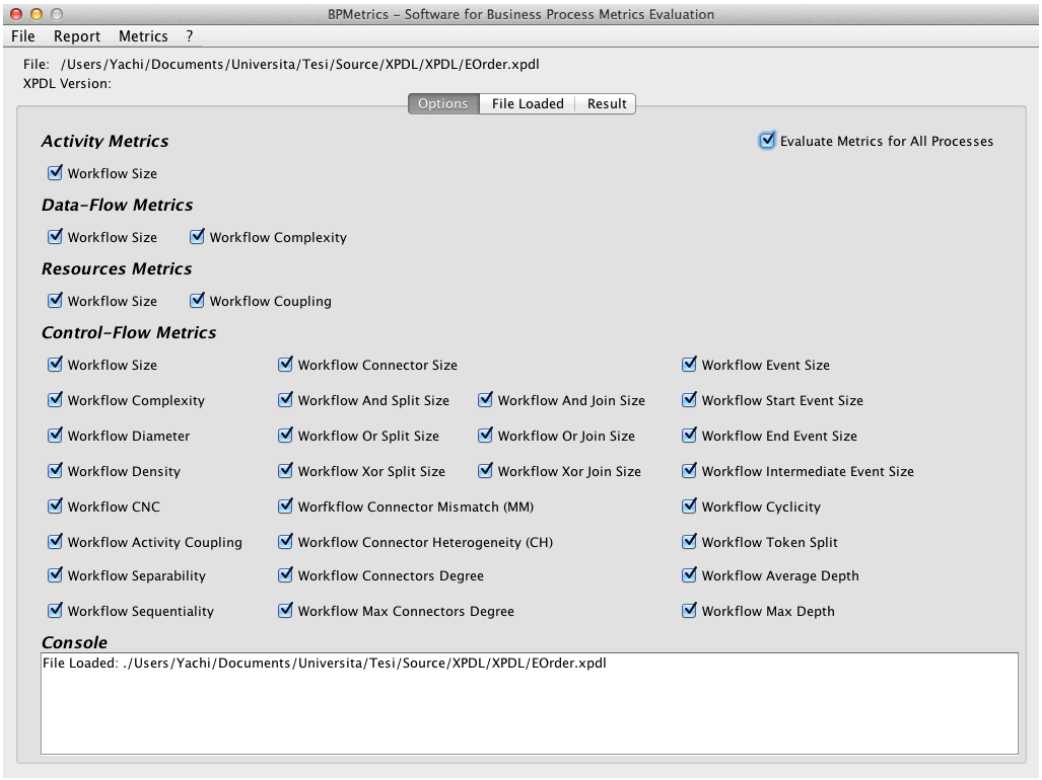
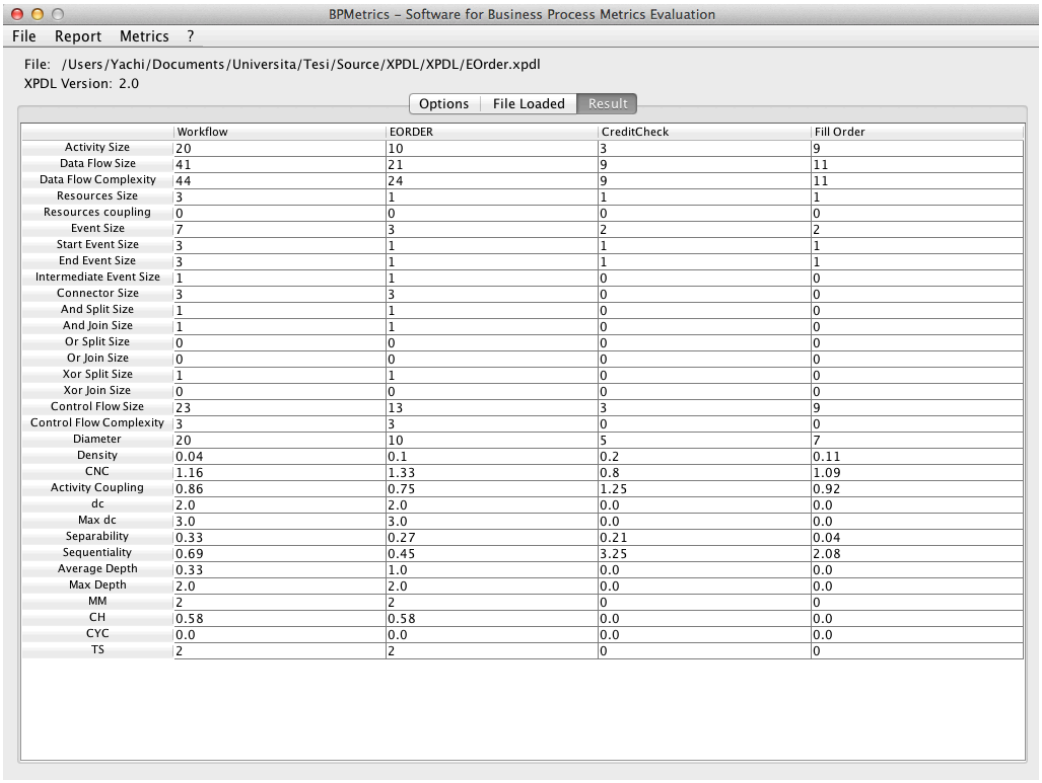


Figure 6.8 Options panel of the BPMETRICS standalone application showing the metrics selected to be computed for the EOrder sample process.

The values of the metrics obtained, as shown in Figure 6.9, will now be analyzed.



	Workflow	EORDER	CreditCheck	Fill Order
Activity Size	20	10	3	9
Data Flow Size	41	21	9	11
Data Flow Complexity	44	24	9	11
Resources Size	3	1	1	1
Resources coupling	0	0	0	0
Event Size	7	3	2	2
Start Event Size	3	1	1	1
End Event Size	3	1	1	1
Intermediate Event Size	1	1	0	0
Connector Size	3	3	0	0
And Split Size	1	1	0	0
And Join Size	1	1	0	0
Or Split Size	0	0	0	0
Or Join Size	0	0	0	0
Xor Split Size	1	1	0	0
Xor Join Size	0	0	0	0
Control Flow Size	23	13	3	9
Control Flow Complexity	3	3	0	0
Diameter	20	10	5	7
Density	0.04	0.1	0.2	0.11
CNC	1.16	1.33	0.8	1.09
Activity Coupling	0.86	0.75	1.25	0.92
dc	2.0	2.0	0.0	0.0
Max dc	3.0	3.0	0.0	0.0
Separability	0.33	0.27	0.21	0.04
Sequentiality	0.69	0.45	3.25	2.08
Average Depth	0.33	1.0	0.0	0.0
Max Depth	2.0	2.0	0.0	0.0
MM	2	2	0	0
CH	0.58	0.58	0.0	0.0
CYC	0.0	0.0	0.0	0.0
TS	2	2	0	0

Figure 6.9 Result panel of the BPMETRICS standalone application showing the metric results computed for the EOrder sampe process

6.5.1 Activity Metrics

The only activity metrics that the software evaluates is the Activity Size. The BPMETRICS application returns these results:

- Workflow: 20
- Eorder main process: 10
- Credit Check SubProcess: 3
- Fill Order SubProcess:9

The correctness of the results is now checked by manual calculating.

In the EOrder main process, we have 10 activities (2 of them are sub Process), 3 Routing Task. Hence,

- *Activity Size = 10*

In the Credit Check sub-process, we have 3 Activities, 0 Routing Task. Hence,

- *Activity Size = 3*

In the Fill Order sub-process, we have 9 activities, 0 Routing Task.

- *Activity Size = 9*

6.5.2 Data Flow Metrics

For the Data Flow Metrics the BPMETRICS evaluates 2 metrics: Size and Complexity. The results follow:

Data Flow Size

- Workflow : 41
- EOrder main process: 21
- Credit Check SubProcess: 9
- Fill Order SubProcess:11

Data Flow Complexity

- Workflow: 44
- EOrder main process: 24
- Credit Check SubProcess: 9
- Fill Order SubProcess:11

All these metrics are correct since the Eorder main process activities are using 21 parameters and the process itself has 3 routing tasks. The activities in the Credit Check SubProcess use 9 parameters and the ones in the Fill Order SubProcess use 11 parameters.

6.5.3 Resources Metrics

For the resources Metrics the BPMetrics evaluates 2 metrics: Size and Coupling. The results follow:

Resource Size

- Workflow : 3
- Eorder main process: 1
- Credit Check SubProcess: 1
- Fill Order SubProcess:1

Resource Coupling

- Workflow: 0
- Eorder main process: 0
- Credit Check SubProcess: 0
- Fill Order SubProcess:0

Considering the correctness of the result, Resource size is correct since each process has a single Swim Lane so each process has only one resource. Also the Resource coupling is correct due to the fact that having only one swim lane for each process mean that there aren't transition that are crossing swim lanes.

6.5.4 Control Flow Metrics

The majority of the metrics that the software evaluates are grouped under the Control Flow Metrics category and now we are going to see in details the results:

Event Size:

- Workflow: 7

- Eorder main process: 3
- Credit Check SubProcess: 2
- Fill Order SubProcess:2

Start Event Size:

- Workflow: 3
- Eorder main process: 1
- Credit Check SubProcess: 1
- Fill Order SubProcess:1

End Event Size:

- Workflow: 3
- Eorder main process: 1
- Credit Check SubProcess: 1
- Fill Order SubProcess:1

Intermediate Event Size:

- Workflow: 1
- Eorder main process: 1
- Credit Check SubProcess: 0
- Fill Order SubProcess:0

The four metrics listed above that consider the events are all correct since each process has one start event and one end event and only the EOrder main process has one Intermediate Event.

Now we can consider the Control Flow Metrics related to the connectors inside the processes:

Connector Size:

- Workflow: 3
- EOrder main process: 3

- Credit Check SubProcess: 0
- Fill Order SubProcess:0

AndSplitSize:

- Workflow: 1
- EOrder main process: 1
- Credit Check SubProcess: 0
- Fill Order SubProcess:0

AndJoinSize:

- Workflow: 1
- EOrder main process: 1
- Credit Check SubProcess: 0
- Fill Order SubProcess:0

OrSplitSize:

- Workflow: 0
- Eorder main process: 0
- Credit Check SubProcess: 0
- Fill Order SubProcess:0

OrJoinSize:

- Workflow: 0
- EOrder main process: 0
- Credit Check SubProcess: 0
- Fill Order SubProcess:0

XorSplitSize:

- Workflow: 1
- EOrder main process: 1
- Credit Check SubProcess: 0
- Fill Order SubProcess:0

XorJoinSize:

- Workflow: 0
- EOrder main process: 0
- Credit Check SubProcess: 0
- Fill Order SubProcess:0

All the metrics are correct since only the EOrder main process has three connectors: an AND split, an AND join and a XOR split.
Now we can now consider the Control Flow Size and the Control Flow Complexity and check directly their correctness.

Control Flow Size:

- Workflow: 23
- Eorder main process: 13
- Credit Check SubProcess: 3
- Fill Order SubProcess:9

Control Flow Complexity:

- Workflow: 3
- Eorder main process: 3
- Credit Check SubProcess: 0
- Fill Order SubProcess:0

The control flow size is correct since Eorder main process contains 2 subProcess activities: thus, when evaluating the expanded main process, we have to subtract two activities from the count. However, the Control Flow Complexity is correct since:

$$\boldsymbol{CFC_{AND-split}(rt) = 1}$$

$$\boldsymbol{CFC_{XOR-split}(rt) = fan-out(rt) = 2}$$

$$\boldsymbol{CFC_{OR-split}(rt) = 2^{fan-out(rt)} - 1 = 0}$$

$$CFC(rt) = CFC_{AND-split}(rt) + CFC_{XOR-split}(rt) + CFC_{OR-split}(rt)$$
$$= 1 + 2 = 3$$

Diameter:

- Workflow: 20
- Eorder main process: 10
- Credit Check SubProcess: 5
- Fill Order SubProcess:7

This metric is correct since any single process diameter is correct and the expanded diameter too due to the fact we have not to consider the 2 task representing the sub processes.

Considering the metrics that put in relation the number of nodes and the number of arcs:

Density:

- Workflow: 0,04
- Eorder main process: 0,1
- Credit Check SubProcess: 0,2
- Fill Order SubProcess:0,11

CNC

- Workflow: 1,16

- Eorder main process: 1,33
- Credit Check SubProcess: 0,8
- Fill Order SubProcess:1,09

Activity Coupling

- Workflow: 0,86
- Eorder main process: 0,75
- Credit Check SubProcess: 1,25
- Fill Order SubProcess:0,92

dc

- Workflow: 2
- Eorder main process: 2
- Credit Check SubProcess: 0
- Fill Order SubProcess:0

Max dc:

- Workflow: 3
- Eorder main process: 3
- Credit Check SubProcess: 0
- Fill Order SubProcess:0

All the metrics above are correct since the characteristics of each process are:

- Workflow: N = 31, A = 36
- EOrder main process: N = 15, A =20
- Credit Check SubProcess: N = 5, A = 4
- Fill Order SubProcess: N = 11, A = 12

Separability:

- Workflow: 0,33
- Eorder main process: 0,27
- Credit Check SubProcess: 0,21
- Fill Order SubProcess:0,04

This metric is correct since the Eorder main process has 3 cut vertices, the Credit Check SubProcess has 1 cut vertices and the Fill Order SubProcess has 1 cut vertex.

Sequentiality:

- Workflow: 0,69
- Eorder main process: 0,45
- Credit Check SubProcess: 3,25
- Fill Order SubProcess:2,08

Average Depth

- Workflow: 0,33
- Eorder main process: 1
- Credit Check SubProcess: 0
- Fill Order SubProcess:0

Max Depth

- Workflow: 2
- Eorder main process: 2
- Credit Check SubProcess: 0
- Fill Order SubProcess:0

The metrics Sequentiality, Average Depth and Max Depth are correct but the proof for this kind of metrics is not so simple that you can write it in a clear and comprehensive.

MM

- Workflow: 2
- Eorder main process: 2
- Credit Check SubProcess: 0
- Fill Order SubProcess:0

CH

- Workflow: 0,58
- Eorder main process: 0,58
- Credit Check SubProcess: 0
- Fill Order SubProcess:0

The Connector Mismatch metric (MM) and the Connector Heterogeneitymetric (CH) are both correct since only the EOrder main process has three connectors: an AND split, an AND join and a XOR split.

CYC

- Workflow: 0
- Eorder main process: 0
- Credit Check SubProcess: 0
- Fill Order SubProcess:0

The Cyclicity metric is correct since none of the processes have a cycle between its activities.

TS

- Workflow: 2

- Eorder main process:2
- Credit Check SubProcess: 0
- Fill Order SubProcess:0

Finally, also the Token Split metric is correct since only the EOrder main process has 1 AND-split with 3 outgoing transition and no OR-splits.

Chapter 7

Conclusions and Future Works

The thesis aims at developing an application that allows one to compute some metrics for Business Process Models defined by the XPDL language (XML-Process Definition Language).
Some 30 metrics were implemented, providing a tool as much complete as possible.

7.1 Conclusions

During the research about Business Process Metrics and during the development of the BPMETRICS system, it clearly appeared that many organizations are modelling and designing business processes without the support of metrics to question the quality or properties of their business process models. As result, it may happen that simple processes are modeled in a complex and unsuitable way, leading to a poor readability, lower understandability, higher maintenance costs and inefficient execution.
As modern organizations spend a lot of time and resources in creating and maintaining business processes, the importance of business process metrics is becoming increasingly important over time.
The BPMETRICS application is a step towards this direction, offering to the users a tool to support the development, creation and maintenance of business processes.
Nevertheless, during the analysis and the development, some metrics from the literature have been discarded, not just for being unuseful to provide a quality instrument, but for being uncomputable and based only on some internal characteristics of the business process model.

Another aspect that emerged during the implementation of the algorithms is that the adopted metrics are computed for Business Process Models that strongly linked to the graph-based models and therefore, the algorithms implemented to calculate the metrics are simply algorithms implemented on graphs and consequently are strictly coupled to the complexity and to the problems that the graphs can lead to.

Finally, the project confirmed that one single metric is not capable of identifying the entire complexity of a business process model or the presence of errors and bugs within it. Many metrics describe many aspects of BPM and can be used during the design phase in order to make design decisions based on empirical data and not on the intuition, possibly dictated by a limited experience.

7.2 Future Research Directions

Since this research is still in its early days, many features can be proposed to enrich the BPMETRICS system.

A visual tool for the creation of Business Process Models that integrates with the BPMETRICS application would allow the designer to obtain some metrics in "real time" during the design phase.

BPMETRICS could also benefit from the extension of the supported format and language. In fact, the system could be extended to read XPDL files generated by other design software, implementing the standard XPDL, i.e. not only Together Workflow Editor, covering the various dialects of the XPDL. Continuing this direction, one could extend the application to cover languages not directly derived from the XPDL, such as EPC or YAWL [5, 32, 33].

Some other metrics could be implemented, tailoring them from other scientific fields and trying to adapt them to the Business Process Model.

Finally, another possible improvement may combine the metrics to an error prediction model, based on the calculated metrics. This was highlighted, even if partially, by Mendling in [5]: however this kind of application requires a lot of empirical validation of the metrics.

References

[1] Melenovsky, M.J.; *Business Process Management's Success Hinges on Business-Led Initiatives*. Gartner Note G00129411 (2005)

[2] Workflow Management Coalition website. *XPDL Support and Resources*. http://www.wfmc.org/xpdl.html

[3] Antonini, A.; Ferreira, A.M.; Morascan, S.; Pozzi, G.; *Software Measure for Business Process.* In: Proc. II of the 15th East-European Conference on Advances in Databases and Information Systems, Vienna, Austria, September 20 - 23, 2011.

[4] Vanderfeesten, I.; Cardoso, G.; Mendling, G.; Reijers, H.A.; Van der Aalst, W.; *Quality Metrics for Business.* BPM and workflow handbook, Ed. L. Fischer, pp.179-190 (2007)

[5] Mendling, J.; *Metrics for Business Process Models: Empirical Foundations of Verification, Error Prediction, and Guidelines for Correctness*. Springer (2008)

[6] Lee, G.S., Yoon, J.-M.: *An empirical study on the complexity metrics of petri nets*. In: JTC-CSCC: Joint Technical Conference on Circuits Systems, Computers and Communications, 1990, pp. 327–332 (1990)

[7] Lee, G.S., Yoon, J.-M.: *An empirical study on the complexity metrics of petri nets*. Microelectronics and Reliability 32(3), 323–329 (1992)

[8] Nissen, M.E.: *Valuing it through virtual process measurement*. In: Proc. 15th. International Conference on Information Systems, Vancouver, Canada, pp. 309–323 (1994)

[9] Nissen, M.E.: *Knowledge-based organizational process redesign: using process flow measures to transform procurement*. PhD thesis, University of South California (1996)

[10] Tjaden, G.S., Narasimhan, S., Mitra, S.: *Structural effectiveness metrics for business processes*. In: Proceedings of the INFORMS Conference on Information Systems and Technology, May 1996, pp. 396–400 (1996)

[11] Tjaden, G.S.: *Business process structural analysis*. Technical report, Georgia Tech Re- search Corp. (June 2001)

[12] Morasca, S.: *Measuring attributes of concurrent software specifications in petri nets*. In: METRICS '99: Proceedings of the 6th International Symposium on Software Metrics, Washington, DC, USA, pp. 100–110. IEEE Computer Society Press, Los Alamitos (1999)

[13] Latva-Koivisto, A.M.: *Finding a complexity for business process models*. In: Research report, February 2001, Helsinki University of Technology (2001)

[14] Reijers, H.A.: *A cohesion metric for the definition of activities in a workflow process*. In: Proceedings of the Eighth CAiSE/IFIP8.1 International Workshop on Evaluation of Modelling Methods in Systems Analysis and Design (EMMSAD 2003), pp. 116–125 (2003)

[15] Cardoso, J.: *About the complexity of teamwork and collaboration processes.* In: 2005 IEEE/IPSJ International Symposium on Applications and the Internet Workshops (SAINT 2005 Workshops), 31 January - 4 February 2005, Trento, Italy, pp. 218–221. IEEE Computer Society Press, Los Alamitos (2005)

[16] Cardoso, J.: *Evaluating Workflows and Web Process Complexity*. In: Fischer, L. (ed.) Workflow Handbook 2005, pp. 284–290. Future Strategies, Inc., Lighthouse Point (2005)

[17] Canfora, G., Garc´ıa, F., Piattini, M., Ruiz, F., Visaggio, C.A.: *A family of experiments to validate metrics for software process models*. Journal of Systems and Software 77(2), 113–129 (2005)

[18] Balasubramanian, S., Gupta, M.: *Structural metrics for goal based business process design and evaluation*. Business Process Management Journal 11(6), 680–694 (2005)

[19] Gruhn, V., Laue, R.: *Adopting the cognitive complexity measure for business process models*. In: Yao, Y., Shi, Z., Wang, Y., Kinsner, W. (eds.) Proceedings of the Firth IEEE International Conference on Cognitive Informatics, ICCI 2006, July 17-19, Beijing, China, pp. 236–241. IEEE Computer Society Press, Los Alamitos (2006)

[20] Gruhn, V., Laue, R.: *On experiments for measuring cognitive weights for software control structures*. In: Zhang, D., Wang, Y., Kinsner, W. (eds.) Proceedings of the Six IEEE International Conference on Cognitive Informatics, ICCI 2007, August 6-8, Lake Tahoe, CA, USA, pp. 116–119. IEEE Computer Society Press, Los Alamitos (2007)

[21] Vanderfeesten, I., Mendling, J., Reijers, H., van der Aalst, W., Cardoso, J.: *On a quest for good process models: The cross-connectivity metric*. In: Bellahse`ne, Z., Le´onard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, Springer, Heidelberg (2008))

[22] Conte, S.D.; Dunsmore, H.E.; and Shen, V.Y. (1986). *Software Engineering Metrics and Models*, Benjamin/Cummings Publishing Company, Inc..

[23] Shepperd, M. (1993). *Software Engineering Metrics Volume I: Metrics and Validations*, McGraw-Hill.

[24] Brandes, U., and Erlebach, T., editors (2005). *Network Analysis: Methodological Foundations* [outcome of a Dagstuhl seminar, 13-16 April 2004], volume 3418 of Lecture Notes in Computer Science. Springer-Verlag.

[25] Reijers, H. A., and Vanderfeesten, I.T.P. (2004). *Cohesion and Coupling Metrics for Workflow Process Design*. In J. Desel, B. Pernici, and M. Weske, editors, Proceedings of the 2nd International Conference on Business process Management (BPM 2004), Lecture Notes in Computer Science volume 3080, pp. 290-305, Springer-Verlag, Berlin.

[26] Cardoso, J., Mendling, J., Neumann, G., Reijers, H.A.: *A Discourse on Complexity of Process Models*. In: Eder, J., Dustdar, S. (eds.) BPM Workshops 2006. LNCS, vol. 4103, pp. 117–128. Springer, Heidelberg (2006)

[27] Briand, L.C; Morasca, S.; Basili, V.R.; *Property-Based Software Engineering Measurement*. IEEE Transactions on Software Engineering, VOL. 22, NO. 1, January 1996.

[28] Nissen, M.E.: *Redesigning reengineering through measurement-driven inference*. MIS Quarterly 22(4), 509–534 (1998)

[29] Reijers, H.A., Vanderfeesten, I.T.P.: *Cohesion and coupling metrics for workflow process design*. In: Desel, J., Pernici, B., Weske, M. (eds.) BPM 2004. LNCS, vol. 3080, pp. 290–305. Springer, Heidelberg (2004)

[30] Henry, S., Kafura, D.: *Software structure metrics based on information-flow*. IEEE Transactions On Software Engineering 7(5), 510–518 (1981)

[31] Workflow Management Coalition: *Workflow Process Definition Interface – XML Process Definition Language*. Document Number WFMC-TC-1025, December 5, 2008, Version 2.1b, Workflow Management Coalition (2008)

[32] Mendling, J.; Moser, M.;Neumann, G.: *Transformation of yEPC Business Process Models to YAWL*. In: 21st Annual ACM Symposium on Applied Computing, Dijon, France, vol. 2, pp. 1262–1267. ACM, Dijon, France (2006)

[33] Mendling, J.; Nu¨ttgens, M.: *Exchanging EPC Business Process Models with EPML*. In: Nu¨ttgens, M., Mendling, J. (eds.) XML4BPM 2004, Proceedings of the 1st GI Workshop XML4BPM – XML Interchange Formats for Business Process Management at 7th GI Conference Modellierung 2004, Marburg, Germany, March 2004, pp. 61–80 (2004)