Some Quality Evaluations for Business Processes

Alessio Antonini[1], Alexandre Ferreira[1], Sandro Morasca[2], and Giuseppe Pozzi[1]

[1] Politecnico di Milano, P.za L. da Vinci 32 I-20133 Milano -Italy-
[2] Università degli Studi dell'Insubria, via Carloni 78 I-22100 Como -Italy-
ferreira@elet.polimi.it sandro.morasca@uninsubria.it
giuseppe.pozzi@polimi.it

Abstract. Designing a business process, which is executed by a Workflow Management System, recalls the activity of writing software source code, which is executed by a computer. Different business processes may have different qualities, e.g., size, structural complexity, some of which can be measured based on the formal descriptions of the business processes. This paper defines measures for quantifying business process qualities by drawing on concepts that have been used for defining measures for software code.
Specifically, the measures we propose and apply here to business processes are related to attributes of *activities*, *control-flow*, *data-flow*, and *resources*. This allows the business process designer to have a comprehensive evaluation of business processes according to several different attributes.

1 Introduction

A *workflow schema*, also know as *graph* or *process model*, is the formal description of a business process (BP), where single atomic work units (*task*) are coordinated, scheduled and assigned to processing entities (*agent*) to achieve a common goal. Examples of BPs are processes related to car rentals, insurance claims, bank loans etc.

A workflow schema may be instantiated several times, and each instance (called *case*) may be executed with its own specific data, e.g., customer's name, preferred car type, rental date.

Agents can be a software application (e.g., a database system), a human (e.g., a customer representative) or a combination of both (e.g., a human using a software program). A workflow management system (WfMS) is a complex software system [1] that fully takes over the responsibility for the coordinated execution of tasks and cases, respecting defined constraints, deadlines, requirements [2]: a WfMS relies on a database management system (DBMS) for data management.

Workflow schemata can be graphically described by several different formalisms: some of them come from research groups [3, 4], including extensions of Petri Nets [5], while others come from no-profit organizations which collect experts of the field [6, 7]. In addition to the process model, other models can be defined to enrich the formal description of the BP. Among them, we here mention the data/information model [3], the organizational model [8], the exception model [9], and the transaction model [10].

The activity of defining a business process may be considered similar to the activity of writing source code in a programming language, where requirements must be collected, analyzed, and implemented. Some Software Engineering measures apply to the activity of defining a business process, too. Software Engineering measures typically aim at quantifying the complexity, size, volume, of a software product for predicting its number of bugs, robustness, safety, and development and maintenance costs. For instance, there may be a significant correlation between the cyclomatic number of a software code and its fault-proneness [11].

In this paper, we propose a novel approach to defining several measures for quantifying business process qualities, analyzing the formal description of a business process, and helping designers obtain a broader view of the business process being modeled.

The remainder of this paper is organized as follows. Section 2 provides a survey of the state of the art. Section 3 describes the approach we propose in detail. Section 4 illustrates our approach via an application example and shows the results we obtained. Section 5 summarizes some concluding remarks and future research directions.

2 Related Work

In Empirical Software Engineering, a number of measures are defined for quantifying several software attributes for the coding phase [12] and the other software life cycle activities. Here, according to our goal, we will mainly focus on the measures that are related to the software code. Thus, this section considers the state of the art, mainly focusing on the software measures from a pure Software Engineering approach (Section 2.1) and on process definition measures explicitly referred to business process and workflow management (Section 2.2).

For both approaches, it should be clear that the number of executions of a loop, being it a `while Condition do ...` in a source code or a set of activities belonging to a cycle in a BP, does not affect the evaluation of the complexity of the schema.

2.1 Software Measures

Many hundreds of software measures exist, so we here focus on those that are most closely related to our approach. It is typical in Empirical Software Engineering to divide attributes (i.e., qualities) into internal and external ones. Specifically, internal software attributes are those that can be measured based only on the knowledge of a software artifact. The literature considers several internal attributes of software, such as *size*, *structural complexity*, *cohesion*, *coupling*, and *length* [13–15]. External software attributes can only be measured if, in addition to the knowledge of a software artifact, knowledge is available about the environment of the software artifact and of the interactions between the software artifact and the environment. For instance, *maintainability*, i.e., the ease of maintaining a software artifact, is an external software quality, as it is is likely to be lower for the authors than for other people or if adequate tool support is available when carrying out maintenance. Internal software attributes have no interest *per se*, but they may be easy to measure, while external software attributes are the interesting ones from a practical point of view, but they may be much more difficult to measure. For a discussion on internal and external software attributes, see [16].

The most relevant qualities for our approach are internal ones, namely, *size*, *structural complexity*, and *coupling*. When defining a new measure for an attribute, it is necessary that one first make sure that the measure makes sense, i.e., that it complies with a few properties that may be expected of measures for that attribute. The approach of [14, 15] defines properties, called axioms, which are introduced via a graph-based representation of a software artifact, called *system*. We now concisely summarize this approach, by referring to the simple example in Figure 1, which shows a system S composed of two *modules* M_1 and M_2. E_1 and E_2 are the *elements* of M_1 and M_2, respectively, and are connected by a *relationship* R. In general, an element is a node of the graph, a relationship an arc of the graph, and a module a subgraph of a given graph. The axioms introduced for an internal software attribute are necessary properties. So, if a measure does not satisfy them, we exclude that it is an adequate measure for the attribute. If, instead, the measure satisfies them, then we have supporting evidence (though not necessarily sufficient) for the measure to be an adequate measure for the attribute, so we say that the measure is a candidate measure for that attribute, to be on the safe side.

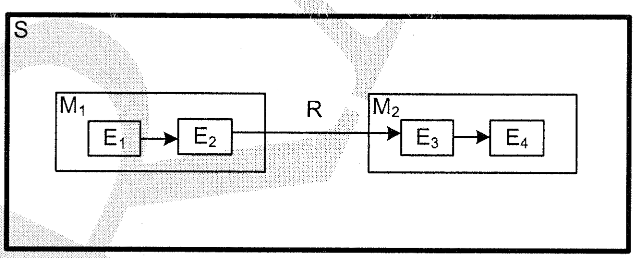

Fig. 1. An example system

We now provide an informal explanation of the axioms of [15] for *size*, *complexity*, and *coupling*, which are the three attributes of interest in this paper.

Size. Considering Figure 1, the two axioms [14, 15] for *size* measures assert that: a) the overall size of S can never be greater than the sum of the sizes of M_1 and of M_2, if every element E_i belong either to M_1, or to M_2, or to both of them; b) the overall size of S equals the sum of the sizes of M_1 and of M_2, if every element E_i belongs either to M_1, or to M_2, but *not* to both of them. A measure is a candidate *size* measure if it complies with all of the above axioms.

Complexity. Considering Figure 1, the two for *complexity* measures assert that: a) the overall complexity of S can never be smaller than the sum of the complexity of M_1 and of M_2, where S is composed by M_1 and M_2; b) the overall complexity of S equals the sum of the complexity of M_1 and of M_2, if M_1 and M_2 are two disjoint modules, with no connections from elements of one to elements of the other. A measure is a candidate *complexity* measure if it complies with all of the above axioms.

Coupling. Considering Figure 1, the four axioms for coupling measures are related to the relationships between the elements from one module and the elements from another module (*coupling*) in S: a) the coupling of a module with no external relationship is null; b) if we add a new relationship R_2 to an existing module M_1, the coupling of M_1 does not decrease; c) if we join the two modules M_1 and M_2, the coupling of the resulting module is never greater the the sum of the coupling of M_1 and of M_2; d) if we join two disjoint modules M_1 and M_2, the coupling of the resulting module is the sum of the coupling of M_1 and of M_2. A measure is a candidate *coupling* measure if it complies with all of the above axioms.

According to the above axioms, the historical approaches of software measures can be reviewed.

One of the most well known source code measures is LOC - lines of code, which simply counts how many lines there are within a source code. This measure is a *size* measure, according to the above axioms of *size*.

The *information flow metric* from Kafura and Henry [17, 18] was introduced as a complexity measure of the flow of information among the modules. This is not a measure forany of the above attributes.

Halstead's Software Science [19] introduces a number of measures for internal software attributes. Based on the number of distinct operators (n_1) and the number of distinct variables (n_2), as well as their total occurrences (N_1 and N_2, respectively), Software Science defines a measure for the *length* (N_1+N_2), *length estimator* ($n_1 \times log_2(n_1) + n_2 \times log_2(n_2)$), *volume* ($(N_1 + N_2) \times log_2(n_1 + n_2)$), and *difficulty* ($n_1/2 \times N_2/n_2$) of the code. The first measure can be considered a *size* measure, as it fulfills all of the above axioms for *size*, but the others do not satisfy any of the above axiom sets for internal software attributes.

McCabe's cyclomatic complexity [20] counts how many linearly independent execution paths can be identified within a source code. Despite its name, this measure is not a *complexity* measure, according to the above axioms, though it may be transformed into one by simply subtracting 1 from it.

2.2 Quality Measures Applied to Process Definitions

A comprehensive review of the existing qualities and their extensions to the analysis of BPs comes from Cardoso et al. in [21–23].

At first, LOC for a BP is defined by considering the number of activities (NOA), the number of activities and of control flow elements (NOAC), and the number of activities, joins, and splits (NOAJS) within a process definition. These are *size* measures, according to the above axioms.

The extension of the *information flow metric* from Kafura and Henry by Cardoso et al. is defined as $length(M) \times (fan-in(M) \times fan-out(M))^2$. The resulting measure does not fulfill any of the above axioms sets.

The extension of Halstead's *length estimator* by Cardoso et al. (Halstead-based Process Complexity - HPC) takes n_1 as the number of activities, splits, joins, and control structures, n_2 as the number of workflow variables or constants used by the BP, N_1 as the overall number of occurrences of activities or of control structures, N_2 as the overall number of occurrences of workflow variables or constants. Other measures, like *volume* and *difficulty* are regularly computed as originally defined by Halstead. The resulting measures *length estimator*, *volume*, and *difficulty* of HPC do not fulfill any of the above axiom sets.

By using McCabe's cyclomatic complexity, Cardoso et al. compute the cyclomatic number for a graph G by Mills' theorem as: $V(G) = D + 1$, where D is the number of control structures. If a control structure has n outgoing arcs, its D value is $n-1$. It can be easily observed that this measure does not fulfill the above axiom set for *complexity*. The authors then propose to compute the cyclomatic complexity as D, and identify the contribution from every control structure a to the overall measure (control flow complexity - CFC) as: $CFC_{AND-split}(a) = 1$, $CFC_{OR-split}(a) = 2^{fan-out(a)} - 1$, $CFC_{XOR-split}(a) = fan-out(a)$. The overall CFC of a graph is the sum of $CFC_{AND-split}$, $CFC_{OR-split}$, $CFC_{XOR-split}$ for any existing control structure of that graph. This measure fulfills all the above axioms for *complexity*.

Gruhn and Laue [24] extend the analysis of Cardoso, including the concept of *nesting depth* and *jumps out a control structure*. The authors assert that a greater nesting depth and the not well structuredness imply greater complexity, while actually higher levels of nesting depth and spaghetti coding decrease the readability of the business process. The authors also introduce some empirical evaluations of the cognitive weight for basic control structures, tailoring the weight to every control structure of a business process.

3 Our Approach

The measurements from Cardoso et al. (see Section 2.2) may not suffice to consider **all** the facets featured by the graph of a BP. We here introduce measures for attributes of *activity*, *control-flow*, *data-flow*, and *resource*. This enables us to compute a wider set of parameters for every graph describing a BP.

In what follows, we shall refer to a BP as a formally defined oriented graph, including a finite set of nodes (N) and the relationships which define the flow

(F) of the process, where $F \subseteq N \times N$. Thus the process p is defined as: $p =< N, F >$. A node can be a task (T) or a split/join (routing task - RT): obviously, $T \cap RT = \emptyset$.

3.1 Activity Attributes

The *activity attributes* are about the activities that compose a BP. Here, we define a size measure.

As a reference, we assume that the size of a task equals one, i.e. $Size_A(task) = 1$. The graph of a BP may include some supertasks (aka subflows or subprocesses): the $Size_A(supertask)$ is the sum of the size of the n tasks belonging to that supertask, i.e. the number (n) of tasks included into the supertask.
Thus, if a process p has n_T tasks - which are not included into any supertask - and n_{ST} supertasks, the overall size is:

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

It can be easily verified that such a measure is a *size*, as all the axioms concerning size are fulfilled.

Proof. We have to consider the two axioms of Section 2.1 which hold for a *size* measure.

With respect to the first axiom, we consider the process $p =< E_p, R_p >$ made by E_p elements and R_p relationships (arcs). We split p in two subprocesses p_1 and p_2. We have that

$$p' = p_1 \cup p_2$$

where $p_1 = < E_{p_1}, R_{p_1} >$ and $p_2 = < E_{p_2}, R_{p_2} >$.

If $p_1 \cap p_2 \neq \emptyset$, then p_1 is made of R_{p_1} elements, and p_2 is made of $R_p - R_{p_1} + 1$; consequently, p' is made of $R_p + 1$ elements, implying that $p \subset p'$. The first axiom holds.

With respect to the second axiom, we still consider the process $p =< E_p, R_p >$ made by E_p elements and R_p relationships (arcs). We split p in two subprocesses p_1 and p_2. We have that

$$p' = p_1 \cup p_2$$

where $p_1 = < E_{p_1}, R_{p_1} >$ and $p_2 = < E_{p_2}, R_{p_2} >$.

If $p_1 \cap p_2 = \emptyset$, then p_1 is made of R_{p_1} elements, and p_2 is made of $R_p - R_{p_1}$; consequently, p' is made of R_p elements, implying that $p = p'$. The second axiom holds, too. □

3.2 Control-Flow Attributes

The *control-flow attributes* are defined based on the pure static structure of the graph of a BP. We introduce a complexity measure and a size measure.

Our approach adheres to the previous work from Cardoso et al [21–23]. As in Section 2.2, we define a *control flow complexity* for any of the rt routing tasks (AND, OR, XOR split): $CFC_{AND-split}(rt) = 1$; $CFC_{OR-split}(rt) = 2^{fan-out(rt)} - 1$; $CFC_{XOR-split}(rt) = fan-out(rt)$. The overall *control flow complexity* (CFC) for a process p is the sum of the complexities originated by the splits as:

$$\begin{aligned} Complexity_{CF}(p) = & \sum_{rt \in AND-split} CFC_{AND-split}(rt) + \\ & + \sum_{rt \in OR-split} CFC_{OR-split}(rt) + \\ & + \sum_{rt \in XOR-split} CFC_{XOR-split}(rt) \end{aligned} \tag{2}$$

This measure satisfies all of the axioms for *complexity*.

We also introduce the concept of *size* of the graph of a BP, defined as the sum of the number of activities (NOA) and the number of activities and control elements ($NOAC$) of a process. This is a size measure, defined as:

$$Size_{CF}(p) = NOA + NOAC \tag{3}$$

All the axioms concerning size are fulfilled: thus, this is a *size* measure.

3.3 Data-Flow Attributes

The *data-flow attributes* address the flow of information among the several activities involved in the graph of a BP. Intuitively, the more information flow among the activities, the higher the resulting complexity.

We can now define the set ($DataFlow$) of data managed by p as the set $v_1, v_2 \ldots v_k$ such that: a) $\forall n \in N$, $Input(n)$ is the set of data received by the node n and it is defined as $V_i^n \subseteq DataFlow$; b) $\forall n \in N$, $Output(n)$ is the set of data produced by the node n and it is defined as $V_o^n \subseteq DataFlow$. Since a routing task (RT) can only read data, *Output* can be associated to normal tasks (T), only.

We then consider four kinds of data managed by a process p:

i. Reference: these data (DR) univocally identify a process instance (e.g., customer_Id, student_Id, reservation_Id). In general, the path followed by these data is the control flow of the graph of a BP;
ii. Operational: these data (DO) are needed by an activity for its processing;

iii. Decision: these data (DD) are a subset of the operational data ($DD \subseteq DO$) and are used by routing tasks (RT) to selectively activate the outgoing/incoming arcs of the graph;
iv. Contextual: these data (DC) belong to a wider category of data, are relevant for the business process, and can be used both as input and as output.

As a consequence, we can define the *DataFlow* as: $V^n_{i,o} = \{DR^n_{i,o} \cup DOi, o^n \cup DC^n_{i,o}\}$. We can now define an attribute which considers the amount of data (number of data items) effectively managed by the process p, where n_T is the number of the activities of the process.

$$Size_{DF}(p) = \sum_{j=1}^{n_T} V^j_{i,o} \tag{4}$$

The measure $Size_{DF}$ for a process p is a *size*, since it fulfills all the axioms for size.

We can also define another measure, which relates to *complexity*: such a measure, takes into consideration both a component deriving from the routing tasks and a component deriving from the tasks which set up a BP. Since routing tasks (RT) have a lower complexity if compared with normal tasks (T), we can assume that the complexity of *DataFlow* for a RT is: $Complexity_{DF}(RT) = 1$. On the other hand, the complexity of *DataFlow* for a task T is: $Complexity_{DF}(T) = V^t_o$. Thus, the resulting overall complexity for data flows is the sum of the complexities of the two components:

$$\begin{aligned} Complexity_{DF}(p) = & \sum_{j=1}^{n_{RT}} Complexity_{DF}(RT_j) + \\ & + \sum_{j=1}^{n_T} Complexity_{DF}(T_j) \end{aligned} \tag{5}$$

The measure $Complexity_{DF}$ for a process p is a *complexity*, since it fulfills all the axioms for complexity.

3.4 Resource Attributes

The *resource attributes* consider the resources required by the graph of a BP and used during process execution. If we assume to have R resources available for the execution of the n_T activities of a process p, we may define a size measure as:

$$Size_R(p) = \sum_{i=1}^{n_T} r_i = R \tag{6}$$

This measure is very similar to the parameter NOA of Formula 3: again, all the axioms concerning size are fulfilled and the measure is a *size*.

We can also define a coupling measure for resources. This measure is strictly based on the BPMN notation [7] used to graphically depict a BP. We consider the number of arcs which cross two (or more) swim lanes: every crossing means that the work item requires a new (different from the previous one) resource for its execution. We thus define a *coupling* measure as:

$$Coupling_R(p) = H \tag{7}$$

where H is the number of arcs which cross at least two swim lanes. The measure $Coupling_R$ for a process p is a *coupling*, since it fulfills all the axioms for coupling.

Proof. With respect to Figure 1, we consider the process S made by two modules M_1 and M_2, one module corresponding to one lane, only. The modules are connected via $r \neq 0$ relationships, where $r \in \mathcal{N}$. Let H be the sum of the relationships between M_1 and M_2: hence, $Coupling_R(S) = H$. The coupling of the two separate modules is: $Coupling_R(M_1) = H_1$; $Coupling_R(M_2) = H_2$.

We have to consider the four axioms of Section 2.1 which hold for a *coupling* measure.

With respect to the first axiom, if M_1 and M_2 have no external relationship, then $Coupling_R(p) = 0$. The coupling of a module with no external relationship is zero: the axiom holds.

With respect to the second axiom, if we assume to add a new relationship to M_1, the new value for the coupling will become $Coupling_R(M_1) = H_1 + 1$.

We define as $OuterR(m)$ the set of the relationships (arcs) outgoing from a module m. Thus, if M_2 is a subset of M_1 such that $< E_3, E_4 >=< E_1, E_2 >$, where E are the elements of the two modules, and $OuterR(m_2) \supseteq OuterR(M_1) \wedge R_2 \supseteq R_1$, then

$$Coupling_R(M_2) = H_2 + H_1 + 1 \geq H_1 + 1 = Coupling_R(M_1)$$

and the axiom holds.

With respect to the third axiom, if M_1 and M_2 share r relationships, the resulting coupling is $Coupling_R(M_1 \cup M_2) = H_1 + H_2 - r$. Thus,

$$Coupling_R(M_1 \cup M_2) = H_1 + H_2 - r \leq H_1 + H_2 = Couplig_R(M_1) + Coupling_R(M_2)$$

and the axiom holds.

With respect to the fourth axiom, if M_1 and M_2 do not share r relationships, then $M_1 \cap M_2 = \emptyset \wedge OuterR(M_1) \cap OuterR(M_2) = \emptyset$. Thus, $r = 0$, and

$$Coupling_R(M_1 \cup M_2) = H_1 + H_2 = Couplig_R(M_1) + Coupling_R(M_2)$$

and the fourth axiom holds, too. □

4 Application Example

In this Section we introduce a reference process, and evaluate for it all the measures we defined in Section 3.

4.1 Business Process of Reference

As process of reference, we assume the "Sale Order" business process graphically depicted by Figure 2 according to the BPMN notation [7]. The agent Sale Manager (topmost lane - Figure 2) receives the purchase order (Receive Order) from the customer and checks with the Finance Department (subprocess Check Finance) if the payment has been received. The order can be declined (Decline Order) or processed (Quantity Check, Quality Check). If the good is not in stock, the agent Production Planner (mid lane - Figure 2) plans the suitable production and waits for it (subprocess Produce): as soon as the supertask Produce is complete, the Sale Manager is informed and he/she can inform the customer (subprocess Notify Full Shipment). Finally, the agent Shipping Operator can complete the process (subprocess Ship and Report).

The four subprocesses are quite simple: Check Finance is made by one task, namely Check Credit, and it is executed by the agent Finance Department; Notify Full Shipment is made by one task, namely Notify Customer, and it is executed by the agent Customer Support; Produce is made by one task, namely Assemble Good, and it is executed by the agent Manufacturing Department; Ship and Report includes two tasks, namely Ship and Delivery Report, and it is executed by the agent Shipping Operator.

Next, if we consider the data managed by the process (workflow data), we find the following workflow vars: CustomerName, ProductName, OrderedQuantity, NumberOfItemsInStock, StockQuantityStatus, StockQualityStatus, NumberOfItemsToShip, NumberOfItemsToProduce, NumberOfProducedItems.

All in all, the process has 16 tasks: 7 simple tasks, 4 complex tasks (subprocess) with a total of 5 tasks, and 5 routing tasks. The process has 3 main swim lanes and 4 resources in the subprocesses.

4.2 Quality Evaluation

We now apply the measures of Section 3 to the reference process of Section 4.1.

Activity Attributes According to the Formula 1 and to the definitions from Section 3.1, if we consider the process of Figure 2 we obtain $Size_A(SaleOrder) = 1 \times 7 + 5 = 12$.

Control Flow Attributes If we apply the Formulae 2, 3 and the definitions from Section 3.2 to the process of Figure 2, we obtain $Complexity_{CF}(SaleOrder) = 5+0+1 = 6$; since $NOA = 11$ and $NOAC = 16$, we obtain $Size_{CF}(SaleOrder) = 11 + 16 = 27$.

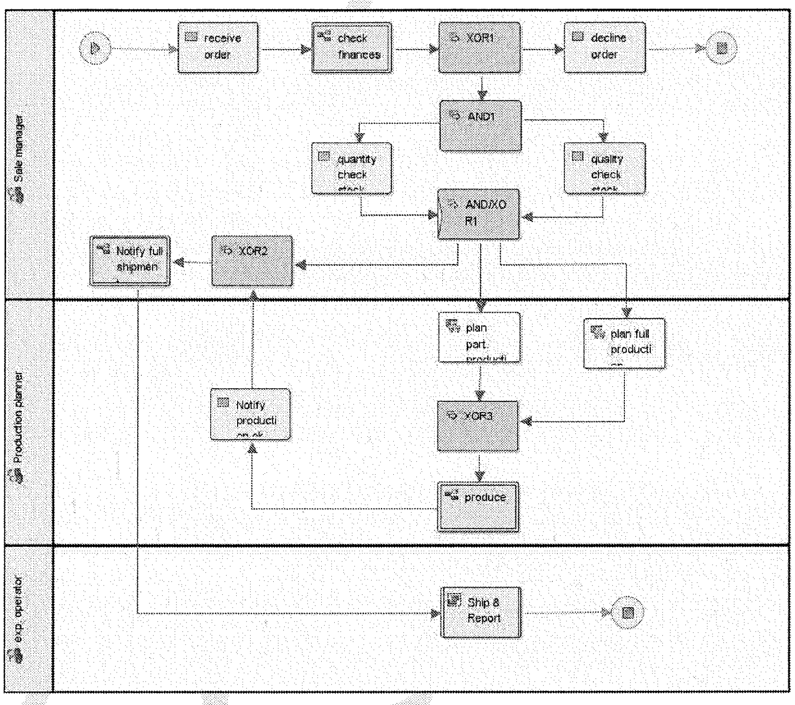

Fig. 2. The Sale Order process, which manages the activities inside an organization when selling goods. The process is graphically described according to the formalism from BPMN, and its code is saved in X-PDL by the Together Workflow Editor tool

Data Flow Attributes By the Formulae 5, 4 and the definitions from Section 3.3, we obtain that the number of workflow vars used as input or output of a task is 48, while 5 workflow vars are used as an input to *split* tasks. Thus, $Size_{DF}(SaleOrder) = 48 + 5 = 53$. On the other hand, for the complexity of the data flow we observe one component coming from the normal tasks and one component coming from the *split* tasks: in our example, we have $Complexity_{DF}(SaleOrder) = 20 + 5 = 25$.

Resource Attributes According to the Formulae 6, 7 and to the definitions from Section 3.4, if we consider the process of Figure 2 we obtain $Size_R(SaleOrder) = 3$ since we have 3 resources involved in the execution of the process: in fact, we have 3 swim lanes, and we do not consider the resources involved in the execution of subprocesses. If we count how many lines (connecting arcs) cross the swim lanes, we obtain $Coupling_R(SaleOrder) = 4$.

4.3 Results

In order to test the approach we propose and applied to evaluate the measures on the process of Figure 2, we consider a simplified version of the same process, where in the topmost lane the activity Quality Check has been removed: consequently, we also removed the routing tasks labeled AND1 and AND/XOR1 - see Figure 3. We can now evaluate the measures of Section 4.2 to the new process of Figure 3.

The new process has 14 tasks: 6 simple tasks, 4 complex tasks (subprocess) with a total of 5 tasks, and 4 routing tasks. The process has 3 main swim lanes and 4 resources in the subprocesses.

Activity Attribute As in Section 4.2, we compute $Size_A(SaleOrder_2) = 1 \times 6 + 5 = 11$.

Control Flow Attribute As in Section 4.2, we obtain $Complexity_{CF}(SaleOrder_2) = 4{+}0{+}0 = 4$; since $NOA = 10$ and $NOAC = 14$, we obtain $Size_{CF}(SaleOrder_2) = 10 + 14 = 24$.

Data Flow Attribute As in Section 4.2, we obtain that the number of workflow vars used as input or output of a task is 44, while 4 workflow vars are used as an input to *split* tasks. Thus, $Size_{DF}(SaleOrder_2) = 44 + 4 = 48$. For the complexity of the data flow, we observe one component coming from the normal tasks and one component coming from the *split* tasks: in our example, we have $Complexity_{DF}(SaleOrder_2) = 18 + 4 = 22$.

Resource Attribute As in Section 4.2, and given that the changes from Figure 2 to Figure 3 do not affect the swim lanes, we obtain $Size_R(SaleOrder_2) = 3$

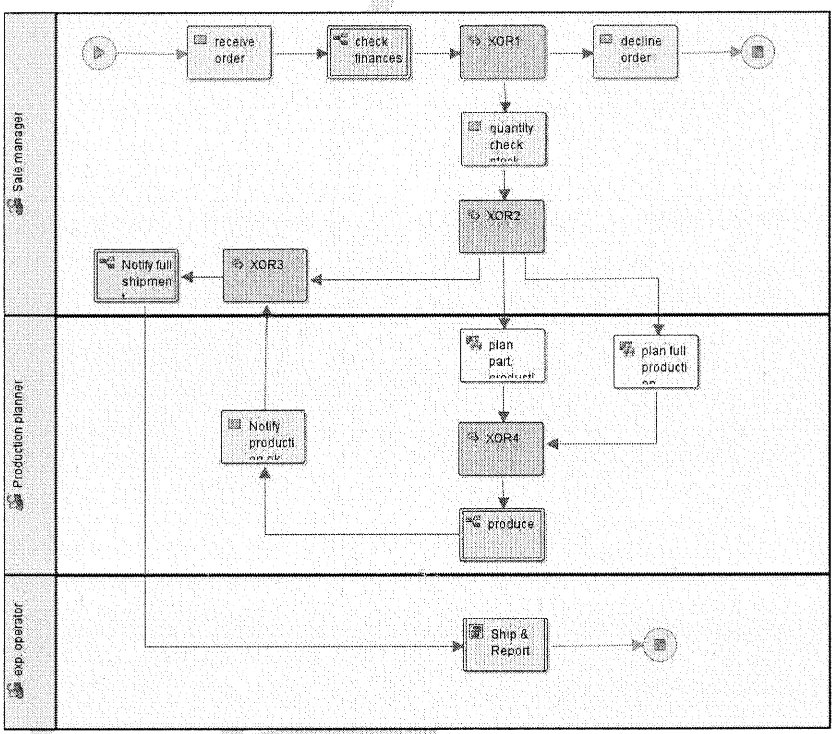

Fig. 3. The business process Sale Order$_2$ is a simplified version of the business process Sale Order

and $Coupling_R(SaleOrder_2) = 4$: as one could easily expect, there is no difference in the resource qualities we evaluate on the two versions of the Sale Order process.

Observation We now compare the measures of complexity and of size for the two processes of Figure 2 (Sale Order) and Figure 3 (Sale Order$_2$). Since Sale Order is heavier than Sale Order$_2$, which we derived from the previous as a simplified version, we expect that the measures we evaluate on the two processes confirm that Sale Order has higher values than the corresponding values of Sale Order$_2$.

We first consider the tuples for the complexity measure, which report the ordered values of $Complexity_{CF}$, and $Complexity_{DF}$. For the two processes, we obtain:

$$\begin{aligned} Complexity(SaleOrder) &= <6, 25> \\ Complexity(SaleOrder_2) &= <5, 22> \end{aligned} \tag{8}$$

Next, we consider the tuples for size, which report the ordered values of $Size_A$, $Size_{CF}$, $Size_{DF}$, and $Size_R$. Again, for the two processes, we obtain:

$$\begin{aligned} Size(SaleOrder) &= <12, 27, 53, 3> \\ Size(SaleOrder_2) &= <11, 24, 48, 3> \end{aligned} \tag{9}$$

As we already outlined in Section 4.3, we do not consider here the *coupling* measure, since both processes present an identical value: $Coupling_R(SaleOrder) = Coupling_R(SaleOrder_2)$.

As we expected, the results confirm that the simplified version of the process (Sale Order$_2$) shows smaller values for any measure both of *complexity* (see the second line of Formula 8) and of *size* (see the second line of Formula 9), if compared with the respective values for the process Sale Order (see the first line of Formula 8 and of Formula 9). This enables us to assert that Sale Order is heavier than Sale Order$_2$.

5 Conclusions

This paper aims at evaluating some measures which are relevant when formally defining a business process. Moving from the software engineering approach in the literature and from previous work on the topic, we defined some axioms according to which we classify the measures as *complexity*, *size*, and *cohesion*. We did not limit our analysis to the pure *control flow* (*control flow attribute*) of the business process: we also considered some other measures, which relate to the activities (*activity attribute*), to the data flow among the activities (*data flow* *attribute*), and to the resources involved in the enactment of the process (*resource attribute*). For all of these facets, we defined the corresponding measure, being it a *complexity*, a *size*, or a *cohesion*.

We considered the BPMN notation to describe the business processes, and saved the process models in the X-PDL file format recommended by the *Workflow Management Coalition* - WfMC. As a first test of our approach, we considered a business process and a lighter version of the same business process: obtained results confirmed that the measures evaluated on the lighter version provide values which are smaller that the homologous values of the original business process.

We also developed a software tool, which reads the X-PDL file format and automatically returns the values of the qualities for that considered business process: thus, we obtain a set of values for every business process.

Future Research Directions The approach we propose has so far been applied to few business processes, and we reported here about one. As future research directions, we plan to consider a much greater number of processes, to further test our approach. Next, the obtained results will be used to check whether there are correlations with other information related to the process, such as costs in developing, running, maintaining the process itself. Particularly, we shall try to correlate the complexity and size of the business process model with the effort required to execute instances of that BP.

We shall also extend the software tool, enabling it to understand other workflow specification languages and other file formats, most of them being proprietary.

Furthermore, the approach can also be used in estimating the additional load to a business process when enriching it to add new functionalities. As an example, expected exceptions of a business process can be mapped inside the graph of that business process by adding suitable tasks [25]: as a result, the enriched business process is heavier. Our approach can be used to estimate the increase of *size*, *complexity*, *cohesion* due to the enrichment in the business process.

Acknowledgements

This research was partially funded by several grants. We also thank the B. Sc. students Mirco Caldera, Stefano Gorla, and the M. Sc. student Duc Xuan Quang VU, who contributed in implementing the running prototype of the system described by the paper.

References

1. Combi, C., Pozzi, G.: Architectures for a temporal workflow management system. In Haddad, H., Omicini, A., Wainwright, R.L., Liebrock, L.M., eds.: SAC, ACM (2004) 659–666

2. Georgakopoulos, D., Hornick, M.F., Sheth, A.P.: An overview of workflow management: From process modeling to workflow automation infrastructure. Distributed and Parallel Databases **3** (1995) 119–153
3. Grefen, P.W.P.J., Pernici, B., Gutierrez, G.S., eds.: Database Support for Workflow Management: The Wide Project. Kluwer Academic Publishers, Norwell, MA, USA (1999)
4. van der Aalst, W.M.P., ter Hofstede, A.H.M.: Yawl: yet another workflow language. Inf. Syst. **30** (2005) 245–275
5. van der Aalst, W.M.P.: Making work flow: On the application of petri nets to business process management. In Esparza, J., Lakos, C., eds.: ICATPN. Volume 2360 of Lecture Notes in Computer Science., Springer (2002) 1–22
6. The Workflow Management Coalition: WfMC - Web Site (2010) http://www.wfmc.org.
7. The Business Process Management Notation: BPMN - Web Site (2010) http://www.bpmn.org.
8. Combi, C., Pozzi, G.: Task scheduling for a temporal workflow management system. In: TIME, IEEE Computer Society (2006) 61–68
9. Casati, F., Ceri, S., Paraboschi, S., Pozzi, G.: Specification and implementation of exceptions in workflow management systems. ACM Trans. Database Syst. **24** (1999) 405–451
10. Grefen, P.W.P.J., Vonk, J.: A taxonomy of transactional workflow support. Int. J. Cooperative Inf. Syst. **15** (2006) 87–118
11. Grady, R.B.: Successful applying software metrics. IEEE Computer **27** (1994) 18–25
12. Pressman, R.S.: Software Engineering: A Practitioner's Approach. McGraw-Hill Higher Education, New York, NY, USA (2001)
13. Briand, L.C., Morasca, S., Basili, V.R.: Property-based software engineering measurement. IEEE Trans. Software Eng. **22** (1996) 68–86
14. Morasca, S.: Measuring attributes of concurrent software specifications in petri nets. In: IEEE METRICS, IEEE Computer Society (1999) 100–110
15. Morasca, S.: Refining the axiomatic definition of internal software attributes. In Rombach, H.D., Elbaum, S.G., Münch, J., eds.: ESEM, ACM (2008) 188–197
16. Morasca, S.: A probability-based approach for measuring external attributes of software artifacts. In: ESEM. (2009) 44–55
17. Kafura, D.G., Henry, S.M.: Software quality metrics based on interconnectivity. Journal of Systems and Software **2** (1981) 121–131
18. Henry, S.M., Kafura, D.G.: Software structure metrics based on information flow. IEEE Trans. Software Eng. **7** (1981) 510–518
19. Halstead, M.H.: Elements of Software Science (Operating and programming systems series). Elsevier Science Inc., New York, NY, USA (1977)
20. McCabe, T.J.: A complexity measure. IEEE Trans. Software Eng. **2** (1976) 308–320
21. Cardoso, J.: How to measure the control flow complexity for web processes and workflows. In Fisher, L., ed.: Workflow Handbook 2005. Future Strategies Inc., Book Division (2005) 199–212
22. Cardoso, J.: Evaluating the process control-flow complexity measure. In: ICWS, IEEE Computer Society (2005) 803–804
23. Cardoso, J., Mendling, J., Neumann, G., Reijers, H.A.: A discourse on complexity of process models. In Eder, J., Dustdar, S., eds.: Business Process Management Workshops. Volume 4103 of Lecture Notes in Computer Science., Springer (2006) 117–128

24. Laue, R., Gruhn, V.: Complexity metrics for business process models. In Abramowicz, W., Mayr, H.C., eds.: BIS. Volume 85 of LNI., GI (2006) 1–12
25. Casati, F., Pozzi, G.: Modeling Exceptional Behaviors in Commercial Workflow Management Systems. In: CoopIS, IEEE Computer Society (1999) 127–138