Plug-in for Annotating Semantic Effect on BPMN Business Process Models

Hilman Nuril Hadi¹, Tri A. Kurniawan², Ismiarta Aknuranda³
Faculty of Computer Science, Universitas Brawijaya Malang, Indonesia
{¹hilmanuril@gmail.com, ²triak@ub.ac.id, ³i.aknuranda@ub.ac.id}

Received 06 August 2019 ; accepted 09 September 2019

Abstract. BPMN has become the standard of business process modeling in describing the existing series of business process. By engaging BPMN, an analyst would possibly able to model the whole business process activities in which he/she may analyze business process upon design time. It can be done by analyzing the structure, behavior, or semantic of process model. In certain condition one may ask a question what would the effects of the process be if it were to be executed up to this point?. However, it cannot be solved with only graphical notation, but with its semantics. In practice, several modeling tools still do not provide a feature for managing information regarding the effects/results in the business process model. In fact, analysts should be supported with a tool in order to semantically enrich a process model with its effects. This article describes effect annotation semantically towards activity in the BPMN model including the rules in representing its effects. The effect annotation will be suited towards activity type (atomic and compound activities). The outcomes of plug-in development of eclipse BPMN2 modeler for representing semantic effect are also described in this paper.

1 Introduction

A business process is a series of activities performed within coordination of organization and technical environment, in which the activities series implementing certain goals [1]. Business process in many real-life organization nowadays has become increasingly more complex. Such complexity is caused by the number of activities in each business process that can be increased at any time. Moreover, the number of business processes in an organization can also increase every time. Thus, it will bring up many challenges (in managing, analyzing, and improving process activity itself). In some cases, an organization there can be hundreds to thousands of business processes to achieve its objectives. For example, Suncorp-Metway (one of companies listed in Australia) has more than 6.000 variances of business process [2]. They committed to increase their business performance by refining their business process continuously [2].

The process knowledge of an organization is usually depicted by its business process models. Many organizations have been modeling their business processes in order to manage their business process in a better manner. This can be proved through research results which report that 93% of 558 business respondents (derived from several industries, organization scope, and geographical region) has conducted business process modeling [3]. However, the business process model is designed not only to define and document business process, but also to store information about the process. Thus, the model can be easily analyzed, updated, and maintained.

Nowadays, there are several business process modeling frameworks that can be used to model a business process, e.g. Flowchart, EPC, UML (notation), BPMN, and
CMM/CMMI. Of course, such frameworks have several differences in modeling a business process, e.g. in terms of concepts and notations. Such differences affect the enterprise in choosing a framework that will be used in the company. Based on several previous reports, the most impressive framework is Business Process Model and Notation (BPMN) which continues to dominate the process standards space and many enterprises use this framework [3], [4], [5]. The primary goal of the BPMN is to provide a notation that is readily understandable by all business users. The BPMN is based on a flowcharting technique tailored for creating graphical models of business process operations so BPMN is easy to use [6]. By engaging BPMN, an analyst would possibly able to model the whole business process activities in which he/she may analyze business process upon design time.

Theoretically, the business process can be analyzed at design time. It can be done by analyzing the structure or behavior of process model, depending on the business process model elements and the relations between its elements [7]. There exist a few number of researches dealing with structure analysis of process model (e.g., [8], [9]). In other hand, an analyst can also analyze the process model based on their semantics, e.g., related to the execution results represented in the form of effects resulting from the execution of one or more activities in a process model [10]. In certain condition one may ask a question what would the effects of the process be if it were to be executed up to this point [11]? However, it cannot be solved with only graphical notation, but with its semantics. In practice, several modeling tools still do not provide a feature for managing information regarding the effects/results in the business process model. In fact, analyst should be supported with a tool in order to semantically enrich a process model with its effects. An analyst is only able to deliver general information on every activity using element ‘annotation’ that has been defined by BPMN. Furthermore, BPMN also possesses some weaknesses [12]. One of weaknesses is BPMN does not provided a feature to describe business process semantically upon effect or outcomes [13]. In order to overcome this problem, Hinge et al (2009) have developed a tool which supports business analyst to provide annotation effect semantically that attached to business process model activity of BPMN [11]. Nevertheless, Hinge et al (2009) do not explain the manner to represent the activity effect easily by considering sentences structure which suitable towards effect specification.

Based on the problem stated above, we develop a plug-in for annotating semantic effect in the BPMN business process model. The plug-in can run on Eclipse BPMN2 Modeler¹. We also develop a framework for representing and annotating effect semantically. The effect annotation will be divided into two parts based on the type of activity, i.e. atomic activity and compound activity. Furthermore, each effect of every activity is represented using Controlled Natural Language (CNL). CNL is selected to enable computer in performing an inference towards the effect representation. The outcomes of plug-in development of eclipse BPMN2 modeler for representing semantic effect are also described in the end of this paper.

Furthermore, the discussion in this article is divided into several sections. Section 2 describes the general description of BPMN and semantically effect annotated BPMN business process model. Section 3 describes the CNL including definition, method, and supporting tools. Section 4 describes the plug-in of representations of effects including plug-in result and plug-in test. Finally, section 5 explains the conclusion of this paper.

2 Semantically Effect Annotated BPMN Business Process Model

Over the past few years, business process management (BPM) has evolved in diverse segments of the computer sciences as a means of managing the complexity of business process. Business Process Model and Notation (BPMN) is a standard for business process modeling that provides a graphical notation to represent all business processes found in organizations, where the business process modeling standards are developed based on traditional flowcharting techniques [6], [14]. BPMN was originally developed by the Business Process Management Initiative (BPMI) which was later taken over by the Object Management Group [15]. BPMN has four categories of elements, such as

¹ The Eclipse BPMN2 Modeler is a graphical modeling tool for authoring business processes. Homepage: https://www.eclipse.org/bpmn2-modeler/
flow objects, connecting objects, swimlanes and artifacts [6], as shown in Figure 1.

**Figure 1.** Categories of BPMN elements, adopted from [6]

In Figure 1, Flow objects are a major part of building a process model in BPMN. The section consists of three core elements, including event, activity, and gateway [6]. Connecting objects are used to link elements of flow objects, which consist of sequence flow, message flow, and association [6]. Swimlanes are used to organize events into different visual categories by separating functionality and responsibility capabilities in a business process model. Artifacts provide basic features to enable the designer process to enrich the process model with additional information. In the element artifacts consist of data object, group, and annotation [6]. Activity on the business process model can be atomic or compound [15]. The atomic activity is an activity that is at the lowest level of detail represented in the diagram and cannot be broken down anymore [16]. While compound activity is a type of activity that is decomposable (can be decomposed to see other levels below) [16]. In the element of activity can be a task (represents atomic activity) or sub-process (represents compound activity) [6].

Furthermore, effect annotation is an annotation that relates to the outcome, result or effect of activity in a business process model [10]. A BPMN business process model can be expanded with the addition of effect annotation on each activity. Effect annotation can be associated with a specific outcome/impact from the execution of activities in a business process model. However, activity in the business process model is associated with activity/action carried out to achieve the objectives of the business process itself, can be atomic or compound. Every activity (atomic or compound) will be given an annotation of effects which is called an immediate effect. The immediate effect on activity is annotated in the form of Conjunctive Normal Form (CNF). It is intended that an immediate effect can be explained in the form of a set of outcome clauses [13]. Furthermore, each outcome clauses from the immediate effect is represented using Controlled Natural Language (CNL).

Basically, the activity in BPMN can be divided into two types, i.e. atomic activity and compound activity. In BPMN, the two activities are modeled using activity elements, atomic activity using task elements and compound activities using subprocess elements. The different types of activities affect the kind of immediate effect, can be deterministic or non-deterministic. Immediate effect must be unique, where the uniqueness of the effect is the result of the implications of the unique activity, for example: activity \((t_1)\) produce one effect \((e_1)\) \((t_1 \rightarrow e_1)\), so that other activities should not define an effect similar to activity \(t_1\), because it is not unique. Effects \((e)\) on atomic activity \((t)\) is deterministic, where the effect produced only consists of one effect scenario \((es)\) [10], [13]. However, effects \((e)\) on compound activity \((t)\) is non-deterministic, where the effect produced may consist of one or more effect scenarios \((es)\) [10], [13]. A compound activity can produce more than one effect scenario because it allows there are a number of execution paths, where the execution path is caused by a path choice that can be traversed (there is an OR or XOR notation). Whereas, an atomic activity only consists of 1 effect scenario. Therefore, the immediate effect on an
activity can be defined as a set of effect scenarios \( \{ e_{s1}, ..., e_{sn} \} \), where 1 and n indicate the number of the execution path. An immediate effect on an activity composed by a set of clauses/phrases that are joined by the AND-gate or OR-gate. AND-gate is used to combine several defined clauses. Whereas, OR-gate is used to separate clauses in a set of immediate effect. For example, the immediate effect of activity is \( \{ c_1 \land c_2 \lor c_3 \land c_4 \land c_5 \} \), so there are two effect scenarios which are indicated by the existence of an OR-gate which each effect scenario has a set of clauses, namely \( e_{s1} = \{ c_1, c_2 \} \) and \( e_{s2} = \{ c_3, c_4, c_5 \} \).

Each clause is formed at least consisting of subject, predicate, and object. In order to homogenize the sentence structure in each clause, this rule proposes a passive voice structure as a structure used to represent a clause in the immediate effect. The passive voice structure used in representing the immediate effect is as follows: Subject + auxiliary verb + past participle +/- by (doer of action), example: A message is accepted by an operator. To represent more than one clause, AND-gate can be replaced with the character “space” and the OR-gate can be replaced with the character “||”. We may consider Figure 2 to illustrate an example of BPMN business process model with effect annotation. This process model represents an order pizza process, which was originally represented by an operator make a transaction for ordering pizza. As can be seen, the operator will be able to serve a payment or cancel a transaction. Furthermore, the immediate effect of each activity is described in Table 1.

### Table 1. Description of Each Activity with Immediate Effect

<table>
<thead>
<tr>
<th>No</th>
<th>Activity name</th>
<th>Immediate Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Make Transaction</td>
<td>A transaction is made by an operator.</td>
</tr>
<tr>
<td>2</td>
<td>Order Pizza</td>
<td>A request is accepted by an operator. An request is detected by an operator.</td>
</tr>
<tr>
<td></td>
<td>( Compound activity)</td>
<td>An order is accepted by an operator. An order is accepted by a customer.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>An invoice is accepted by a customer. An invoice is accepted by an operator.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A request is detected by an operator.</td>
</tr>
<tr>
<td>3</td>
<td>Pay Invoice</td>
<td>A request is accepted by an operator.</td>
</tr>
<tr>
<td></td>
<td>( Atomic activity)</td>
<td>A request is detected by an operator.</td>
</tr>
<tr>
<td>4</td>
<td>Cancel Transaction</td>
<td>A transaction is cancelled by an operator.</td>
</tr>
</tbody>
</table>

### 3 Controlled Natural Language

Controlled Natural Language (CNL) is engineered from natural languages that have a specific purpose, such as reducing the ambiguity and complexity of natural language. CNL is a language based on a certain natural language, which is more restrictive about lexicon (dictionary/vocabulary), syntax, and/or semantics, and preserves most of its nature [17]. Natural language is widely used by humans as a medium of communication. Natural language today is the most expressive language of knowledge representation [18]. The language is easy to use and understood by humans. CNL restricts grammar and vocabulary systematically to reduce some of the weaknesses of
natural language. CNL became popular because there are many researchers who have developed several methods for controlling natural language, such as Formalized English (FE), Attempto Controlled English (ACE) E2V, DLT Intermediate Language (DLTIL) and several other methods [17].

In this article, we chose ACE method for developing our plug-in. The main reason is that ACE is very expressive, adopt the natural language and contains a comprehensive language that includes syntax and semantics. [17]. ACE is a controlled natural language, that is precisely defined English, can be translated automatically and unambiguously into First Order Logic (FOL) [19]. ACE has been created as a medium between natural language and formal language. The representation of the ACE sentence will look like a natural language, but the ACE representation is a formal language, in concrete the language represents FOL with natural language syntax. Thus, the ACE sentence is a sentence that is easily understood by humans and machines. The use of a formal language such as ACE will result in a high degree of precision [20]. ACE is supported by several tools, including Attempto Parsing Engine (APE) which translates ACE sentences into Discourse Representation Structures (DRS) and FOL, Attempto Reasoner (RACE), ACE RULES System.

4 Plug-in Development

4.1 Plug-in with Effect Annotation

This section discusses a plug-in as additional features in the Eclipse BPMN2 Modeler plug-in called Annotation Effect. The purpose of this development is to provide a plug-in for annotating semantic effect in the BPMN business process model. This plug-in contain two features for representing the immediate effect semantically in each activity, i.e. atomic activity and compound activity. In addition, this plug-in provides a simple graphical user interface (GUI), so analyst can use it easily (interface, familiar language and provide information if there are input errors). Figure 3 represents the platform architecture for developing this plug-in. In Figure 3, the Eclipse Modeling Platform becomes the main platform for running a BPMN2 Modeler and plug-in effect annotation. The BPMN2 Modeler is a plug-in that provides graphical modeling tools to model business processes. Furthermore, the architecture of the plug-in effect annotation is described in Figure 4. The plug-in effect annotation consists of custom task element and custom sub-process element. In such element, there are additional features to represent the effect in accordance with the characteristic of its activities including check the structure of effect sentences. We use an APE Service for checking the structure of effect sentences in BPMN business process model.

![Figure 3. Eclipse Modeling Platform Architecture with the Extended Plug-in](http://attempto.ifi.uzh.ch/site/docs/ape_webservice.html)
The result of this development is a new element activity for representing immediate effect on atomic activity “called a custom task element” and compound activity “custom sub-process element”. Figure 5 represents the interface and properties of custom task element in our plug-in for representing immediate effect on atomic activity. The new feature/property is represented by a red mark “Task: effects definition” tabulation. In such tabulation, there are three main components such as `textfield immediate effect`, `button check effect`, and `textarea output`. The `textfield immediate effect` is used to represent an immediate effect. The `button check effect` is used to check the structure of immediate effect. Furthermore, the `textarea output` is used to display the check results of immediate effect base on effect representation rules.

**Figure 5.** Interface of Element Custom Task for Representing Immediate Effect
Furthermore, Figure 6 represents the interface and properties of custom sub-process element in our plug-in for representing immediate effect on compound activity. The new feature/property is represented by a red mark “Process: effects definition” tabulation. In such tabulation, there are five main components including button add scenario, remove scenario, textfield effect scenario, button check effect scenarios, and textarea output. The button add scenario and remove scenario are used to determine the number of textfield effect scenario. The textfield effect scenario is used to represent the sentence effect of each scenario. The button check effect is used to check the structure of immediate effect (one or more effect scenarios in compound activity). Furthermore, the textarea output is used to display the check results of immediate effect base on effect representation rules.

![Figure 6. Interface of Element Custom Sub-Process for Representing Immediate Effect](image)

### 4.2 Plug-in Test

#### A. Test Procedure

In this section, we describe some test cases for testing our plug-in. After the development of effect annotation plug-in, we test the plug-in in order to check whether the plug-in behaves correctly and precisely in accordance with effect representation rules. This test can be done in several ways, one of which, functionality test of the plug-in that has been developed. It can be done by testing the output of the plug-in. We tested the plug-in based on new elements in its plug-in, i.e. custom task element and custom sub-process element. We, firstly, determine some test cases, test data and expected results for atomic activity and compound activity, described in Table 3 and Table 4, respectively. Afterward, we test the plug-in by running the plug-in in Eclipse BPMN2 Modeler according to its test cases and test data. Finally, we record the plug-in results and compare with the expected results.

In Table 2, we have made 3 test cases based on the characteristics of atomic activity, e.g. each activity has an immediate effect with one effect scenario, wherein one scenario the effect can consist of several clauses $es_e = \{c_1, c_2, ..., c_n\}$. The first test case checks whether the plug-in can represent the immediate effect with just one clause. Then, the second test case checks whether the plug-in can represent the immediate effect with
more than one clause. Both test cases simulate the effect representation correctly. The last test case checks whether the plug-in can give an error message if there is an error in the effect sentence. Error in the sentence can occur at the writing of subject, predicate or object. We marked the input error in the effect sentence with bold and underline.

Table 2. Test Case for Testing the Custom Task Element

<table>
<thead>
<tr>
<th>No</th>
<th>Test Case</th>
<th>Test Data (Immediate Effect)</th>
<th>Expected results (Output Plug-in)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The task has only one effect</td>
<td>Activity name=accept request</td>
<td>The effect 'A request is accepted by an operator.' is correct.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>effect (e): c1 = A request is accepted by an operator.</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>The task has more than one effect</td>
<td>Activity name=accept request</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Effect (e): c1 &amp; c2 = A request is detected by an operator.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>An order is accepted by an operator.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>If there is a sentence error on</td>
<td>Activity name=accept request</td>
<td></td>
</tr>
<tr>
<td></td>
<td>the effect</td>
<td>Effect (e): c1 &amp; c2 = A request is detected by an operator.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>An order is accepted by an operator.</td>
<td></td>
</tr>
</tbody>
</table>

In Table 3, we have made 3 test cases based on the characteristics of compound activity, e.g. each activity has an immediate effect with one or more effect scenarios. Test cases 1 and 2 describe the effect representation in the sub process can consist of 1 or more scenarios of effect, in which each scenario effect can be represented by one or more clauses. Both test cases simulate the effect representation correctly. The last test case checks whether the plug-in can give an error message if there is an error in the effect sentence (the output will give an error message in the wrong sentence only). We marked the input error in the effect sentence with bold and underline.

Table 3. Test Case for Testing the Custom Sub-Process Element

<table>
<thead>
<tr>
<th>No</th>
<th>Test Case</th>
<th>Test Data (Immediate Effect)</th>
<th>Expected results (Output Plug-in)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sub Process has only one effect</td>
<td>Activity name=Order Pizza</td>
<td></td>
</tr>
<tr>
<td></td>
<td>scenario</td>
<td>Effect (e): ex1(c1 ... c5) = A request is accepted by an operator. A request is detected by</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>An order is accepted by an operator.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>An order is accepted by a customer.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>An invoice is accepted by a customer.</td>
<td></td>
</tr>
</tbody>
</table>

The sign ‘<>’ indicates the point of failure, make sure the word after/before the mark is correct.

p-ISSN: 2540-9433; e-ISSN: 2540-9824
2. **Sub process has more than one effect scenarios**

   **Activity name=Order Pizza**
   
   **Effect (e):** es1(c1 ... c5) 
   || es2(c6 ... c8) = A request is accepted by an operator. A request is detected by an operator. An order is accepted by an operator. An order is accepted by a customer. An invoice is accepted by a customer. ||
   A request is accepted by an operator. A request is detected by an operator. An order is rejected by an operator. An order is rejected by an operator.

   1. The effect 'A request is accepted by an operator.' are correct.
   2. The effect 'A request is detected by an operator.' are correct.
   3. The effect 'An order is accepted by an operator.' are correct.
   4. The effect 'An order is accepted by a customer.' are correct.
   5. The effect 'An invoice is accepted by a customer.' are correct.
   6. The effect 'A request is accepted by an operator.' are correct.
   7. The effect 'A request is detected by an operator.' are correct.
   8. The effect 'An order is rejected by an operator.' are correct.

3. **If there is a sentence error on the effect in effect scenarios**

   **Activity name=Order Pizza**
   
   **Effect (e):** es1(c1 ... c5) 
   || es2(c6 ... c8) = A request is accepted by an operator. A request is detected by an operator. An orders are accepted by an operator. An order is accepted by a customer. An invoice is accepted by a customer. ||
   A request is accepted by an operator. A request is detected by an operator. An order is rejected by an operator. An order is rejected by an operator.

   1. The effect 'A request is accepted by an operator.' are correct.
   2. The effect 'A request is detected by an operator.' are correct.
   3. There's an error. Statement: 'An <> orders are accepted by an operator.'
   4. The effect 'An order is accepted by a customer.' are correct.
   5. The effect 'An invoice is accepted by a customer.' are correct.
   6. The effect 'A request is accepted by an operator.' are correct.
   7. There's an error. Statement: 'A request is detect <> by an operator.'
   8. The effect 'An order is rejected by an operator.' are correct.

   The sign '<>' indicates the point of failure, make sure the word after/before the mark is correct.

### B. Test Result

We summarize our plug-in test results in Table 4 with respect to the test cases, test data and expected results previously described in Table 2 and Table 3. According to the results, all of the elements (custom task element and custom sub-process element) in our plug-in have appropriated with the expected result. For example, in Figure 6, we have illustrated the plug-in test associated with the second test case in Table 3. In such illustration, we test the custom sub-process element with test data (activity name= Order Pizza) and the immediate effect (two effect scenarios). Further, we compared our expected result with our plug-in test result. Finally, the plug-in test result has appropriated with the expected result on second test case in Table 3.
Table 4. Result of Plug-in Test

<table>
<thead>
<tr>
<th>Atomic Activity (Custom Task)</th>
<th>Compound Activity (Custom Sub Process)</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Test Case Validation Status (expected result VS plug-in result)</td>
<td>No Test Case Validation Status (expected result VS plug-in result)</td>
</tr>
<tr>
<td>1 The task has only one effect</td>
<td>Sub Process has only one effect scenario</td>
</tr>
<tr>
<td>(Match) Already appropriate</td>
<td>(Match) Already appropriate</td>
</tr>
<tr>
<td>2 The task has more than one effect</td>
<td>Sub process has more than one effect scenarios</td>
</tr>
<tr>
<td>(Match) Already appropriate</td>
<td>(Match) Already appropriate</td>
</tr>
<tr>
<td>3 If there is a sentence error on the effect in effect scenarios</td>
<td>If there is a sentence error on the effect in effect scenarios</td>
</tr>
<tr>
<td>(Match) Already appropriate</td>
<td>(Match) Already appropriate</td>
</tr>
</tbody>
</table>

5 Conclusion
In this paper, we have explained the plug-in for annotating the BPMN business process model with the semantic effect. The plug-in contains two features to represent the immediate effect on each activity, i.e. custom task element and custom sub-process element. The development of this plug-in refers to the concept of effect annotation and representation that are divided into two parts, i.e. the atomic activity and compound activity. Each activity will produce an immediate effect, i.e. deterministic (for atomic activity) and non-deterministic (for compound activity). The immediate effect is annotated in each activity in the form of CNF (conjunctive normal form) so that the analyst can explain the immediate effect in the form of a set of (outcome clauses). Then, each clause effect can be represented by CNL (with an effect representation rule). The outcomes of plug-in development of eclipse BPMN2 modeler for representing semantic effect has also been presented to show our plug-in runs correspond with its expected result. Finally, we assume that our plug-in can help analyst to represent immediate effect on activity in the BPMN business process model.

References