Context Sensitive Grammar for Composing Business Process Model Variants

Riyanarto Sarno, Widyasari Ayu W., Nurul Fajrin A.
Informatics Department, Institut Teknologi Sepuluh Nopember, Indonesia
riyanarto@if.its.ac.id, widyasari.wibowo11@mhs.if.its.ac.id, nurulfajrin@if.its.ac.id

Djauhar Manfaat, M. Sholikhan Arif, Imam Baihaqi
Naval Architecture Department,Institut Teknologi Sepuluh Nopember, Indonesia
dmanfaat@na.its.ac.id, sholikhan_@na.its.ac.id, hq_by12@na.its.ac.id

***Abstract*— composing business process model variants is an activity to form a new variety of business process models in order to fulfill user needs. Nowadays business process model design has been done in various ways. Many well-known approaches tried to specify variants by different process models or expressed them in terms of conditional branches within the same process model. Those two approaches might lead to error prone and redundancy. Another approaches constitute basic parts of a business process and then prepares a method of editing the model afterward. Although this approach is a flexible way to form a suitable model based on user needs, it might not form a correct variation. A correct variation can be constituted if pattern or rule of variation is well defined and applied during business process model composition. In this paper we proposed a form of contextual modelling concept based on Context Sensitive Grammar approach, which provides a flexible and correct methods to compose business variants process. The result show that Context Sensitive Grammar of a certain process variant can be constituted properly based on formal rules by applying a set of affirm production rules during variant model design.**

Keywords—component; process variant management; contextual modelling; process design methods; context sensitive grammar

I. INTRODUCTION

Process variants modelling becomes very important since it is common for organization to have various combination of task or activity in any given business process. Variants in business process models usually related to user specific needs in order to fulfill business rules and regulations. Therefore conventional modelling, which usually use inflexible approach, building each variant of business process, cannot handle variants modelling adequately. In conventional modelling tools e.g. *Petri Nets Modelling Language* (PNML) or *Business Process Model Notation* (BPMN), business process modelling requires one of two optional approaches to be chosen. The first option is building multiple models to emphasize every variation details of business process, resulting excessive amount of task duplication as the common activities are being executed repeatedly in various business process models. The second one is forming multiple variants into one model with conditional branches to accommodate variants, consequently the complexity of integrated model becomes very high and makes it strenuous to analyze and sustain each variants individually [1, 2]. Meeting these challenges the process designers have to prepare an adequate design which allows multitude changes to be implemented on the model without raising more complexity to it. One of the process modelling advanced approach for handling variants in business process adequately is *Process Variants by Options* (PROVOP). In PROVOP approach, there are change operation methods which are applicable to change original process model into variant specific adapted models [3]. However those changes made could mislead the composition of particular activity in variant model. As an example, the original model has right activities sequence of A – B – C – D – E (shown in Fig. 1).

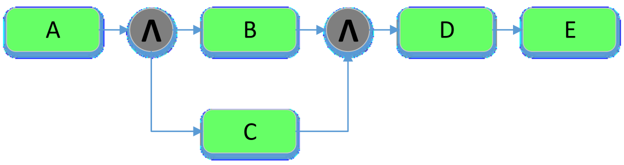

Fig. 1. Example of original model

Using PROVOP we can apply change operation arbitrarily. As an example we move activity C to activity D and vice versa, resulting new variant of activities sequence: A – B – D – C – E as shown in Fig.2.

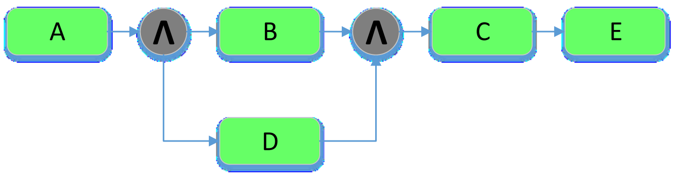

Fig. 2. Example of wrong variant model

This sequence violates the right activities sequence of original process model, yet PROVOP accept this well. Thus we need to define some rule to control change operations in PROVOP. By using *Context Sensitive Grammar* (CSG) we could specify the right activities sequence in variant models. In CSG there are grammars which in every production rule, a non-terminal string is replaced by other non-terminal or terminal string which is at least as long as the first one [4]. Production rules in CSG derive certain conditions to specific

978-1-4799-8386-5/15/$31.00 ©2015 IEEE

consequences. These rules have the flexibility to form any variants models. However, once these rules are implied, every single variant forming has to follow production rule execution. If the condition needed by user to form new variants is not exist, then user is allowed to add new production rule needed.

II. Related Work

A. Process Variants by Options

PROVOP is an advanced approach in business process variant modelling technique. It divides its variant lifecycle into 4 phases [5]. The first phase is process modelling. Modelling process variants must support reuse models. It should be possible to create a new variant by inheriting existing model properties without creating any redundancy or inconsistent model. The next phase is configuration. In configuration process variant accomplished automatically based on specific circumstances. After obtaining a fix model, the model is applied in workflow execution during runtime and maintained its usability for the next process lifecycle.

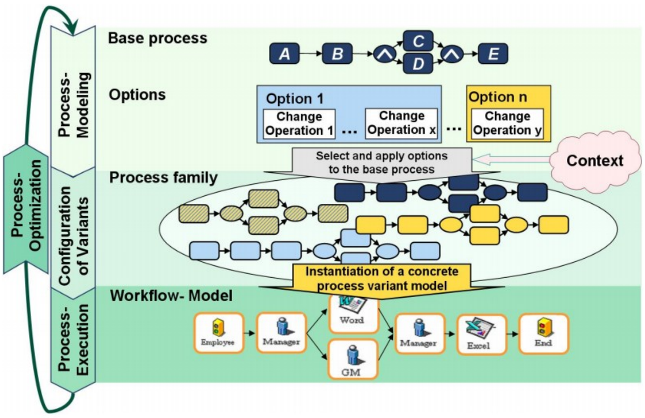

Fig. 3. The PROVOP process variant lifecycle

PROVOP defines its method into three "parts (see Fig. 3): a base process, general model used for forming model variant; change operations, rules used to change base process into model variants; and adjustment points, reference points belong to base process denoting sections in base process which can be changed. There are various ways in PROVOP to design a base process:

1. Base process based on standard process.
2. Base process based on most often used process variant.
3. Base process based on minimum average distance.
4. Base process based on all process variants merging.
5. Base process based on intersection of all process variants

Each base process is equipped with several adjustment points. The extent of adjustment point's addition in base process depends on how many changes need to form new variant through change operation's applications. Change operations, which are applied to base process, will produce new variant from existing model properties. PROVOP can apply move, delete, insert, and modify operation. Using move, we could move process properties from one reference point to another one. Delete is used when we want to remove certain process properties from model, while insert is opposite from delete operation. Modify is used when we want to change process properties data. All change operations can be used in any order, depend on what new process variant like to be produced from change operations sequential execution [3].

B. Context Free Grammar

Context Free Grammar (CFG) is a language generator which derives a certain form of string based on certain rule condition [6]. But the rule itself is free from contextual bound. The derivation notion can be explained with graphical representation called derivation tree. The grammar of natural language in Context Sensitive Grammar are used to constitute and validate produced sentences in the language [7].

A *grammar G* = (*N*, *T*, *R*, *S*) consists of the following components:

1. *N* is a set of finite non-terminal symbols.
2. *T* is a set of finite terminal symbols.
3. *R* which is called production rule, is finite set of $N \times V^*$ where $V = N \cup T$
4. $S \in N$ is a distinguished non-terminal symbol used as starting point of language generator.

In *Context Free Grammar* every production rule is constructed in form of $N \rightarrow V$. Where *V* stand for variables in non-terminal and terminal string or empty string. Derivation of empty string is called empty derivation and takes form of $N \rightarrow \varepsilon$.

Nevertheless, *free grammar* face serious ambiguity problem as shown in Fig. 4. In *Context Free Grammar,* a string could be derived from more than one production rule, resulting commotion in choosing which production rule is necessary to use in certain condition. There are two kinds of ambiguity problem faced by *Context Free Grammar* [8]. *Vertical ambiguity* happens when there is an option between two or more production rule of a non-terminal string.

```
Z  :  A 'y'
   |  'x' B     ;
A  :  'x' 'a'   ;
B  :  'a' 'y'   ;
```

Fig. 4. Example of vertically ambiguous grammar

String *x-a-y* is derived from two options by choosing either first production rule of Z with production rule A or second rule of Z with production rule B.

Horizontal ambiguity (see Fig. 5) happens when there are arbitrary options on the right-hand side of production rule

```
Z  :  'x' A B   ;
A  :  'a'
   |  ε         ;
B  :  'a' 'y'
   |  'y'       ;
```

Fig. 5. Example of horizontally ambiguous grammar

String *x-a-y* is derived from two ways by choosing either Z, first production of A and second production of B or Z, second production of A and first production of B.

C. Context Sensitive Grammar

In Chomsky Hierarchy, Context Sensitive Language is a type-1 language which is also superset of Context Free Language [9] as shown in Fig. 6.

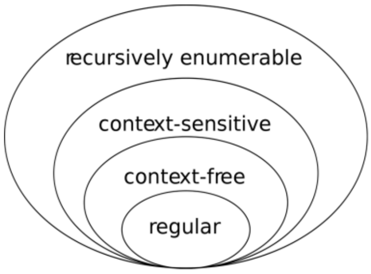

Fig. 6. Chomsky hierarchy

Context Sensitive Language is produced by contextual based *grammar. Context Sensitive Grammar* has the same component as Context *Free Grammar. A sensitive grammar* consist of non-terminal variables, terminal variables, production rules, and a starting point [10, 11]. However, all production rules in *sensitive grammar* are of the form as in (1).

$$\alpha N\beta \rightarrow \alpha\gamma\beta \text{ where } \gamma \text{ is not an empty string} \quad (1)$$

Every string derivation is followed by certain conditions as prerequisite of derived changes. Empty string derivation is not recommended on the right-hand side of any production rule. As consequence as in (2).

$$\alpha \rightarrow \beta \text{ implies } |\alpha| \leq |\beta| \quad (2)$$

The string length in left-hand side of production rule has to be less than or equal the right-side of production rule.

III. The Proposed Method

Our proposed method combines between *Context Sensitive Grammar* and PROVOP approach, resulting variants forming based on specific rule condition. Thus change operations can't be done arbitrarily. We use *Context Sensitive Grammar* to construct original business process model as base process in PROVOP and also to make sure implementation of change operation would generate the right model variant.

A. Context Sensitive Grammar Notation

DEFINITION 1 (*Context Sensitive Grammar*)

$$G_S = (N, \sum, R, BP) \quad (3)$$

Let G_S (3) denotes the *grammar* built *Context Sensitive Language.* The *grammar* consists of *N* the set of non-terminal variables, *N* symbolizes abstract level of sub-model in business process model. Let $MI, MD \in N$ be two types of sub-model abstraction, where *MI* is independent sub-model which could be chosen arbitrarily and *MD* is dependent sub-model whose occurrence depends on *MI* and production rules applied. The set of terminal variables symbolizes as $\sum$. *R* represents the set of production rules and $BP \in N$ is starting point to compose whole base process.

B. Decideability of Context Sensitive Grammar

Context Sensitive Grammar is superior to *Context Free Grammar* when it comes to ambiguity problem. It is clear that derivation on production rules of *Context Sensitive Grammar* are driven by certain characteristics or condition, resulting none of specific derivation string is derived from two or more different production rules.

Vertical and horizontal ambiguous cases can be easily solved by *Context Sensitive Grammar*, since string derivation's condition is explicitly implied in production rule. Thus we will know exactly under what circumstances to apply a certain production rule in string derivation. In *Context Sensitive Grammar* one string derivation in terminal level is produced from exactly one condition of contextual production rule.

```
Z      :     A 'y'         :
A 'y'  :     'x' 'a' 'y'   :
```

Fig. 7. Example of production rule in CSG

String *x-a-y* in production rule (Fig. 7) is produced by exactly one production rule A 'y'. Therefore ambiguity problem would not arise in *Context Sensitive Grammar.*

C. Implementing Provop in Context Sensitive Grammar

DEFINITION 2 (Process Change and Process Variant) [12]

Let *P* denotes the set of possible process schemes while *MI* denotes the set of possible independent abstract sub-model and *MD* denotes the set of possible dependent abstract sub-model whose occurrence depends on *MI. MI, MD* $\in$ *P.* Let *C* the set of possible changes applied to independent sub-model *MI.* Let $I, I' \in MI$ be two independent sub-model schemes and D, $D' \in MD$ be two dependent sub-model schemes. Let $\gamma \in C$ *be* a process change applied only to sub-model independent *MI.* Then:

- *I[γ >I' iff γ is applicable to I and I' is the <u>sub-model independent</u> schema resulting from application of γ to I. I' is variant of I.*
- *I → D, I[γ>I' →D' iff occurrence of D, <u>sub-model dependent</u> schema, depends on I and application of γ to I resulting change on D becomes D'. D' is variant of D.*

DEFINITION 3 (Base Process)

Let *P* denotes the set of possible process schemes while t $\gamma \in C$ *be* a process change applied only to sub-model independent *MI.* Then:

- *A <u>base process</u> BP = (MI, MD, E, AP…) ∈ P is a process model with sub-model independent MI and sub-model dependent MD. E is the set of edges connecting sub-models in the model. A base process is associated with adjustment points AP ⊆ Identifiers x MI x {pre, post}. Therefore adjustment points ap correspond to the entry (pre) or exit (post) point of node n ∈ MI.*

D. School Registration Business Process: A Study Case

School registration is an activity to recruit new student on new term of school year. Each year a school hold a selection system for hundreds participants. The selection program itself is vary between one schools to another, depends on school registration participant's condition at that time.

A register module usually divided into three parts: pre-register activity, additional activity, and register activity. In this case, we would like to use pre-register activity as a study case for this paper. Pre-register activity mostly contains verification activities of student domicile area, graduation year of previous education, graduation type of previous education, and student previous school area. From those verification activities a school can decide whether participants / students could do their pre-register activity independently by using online registration system, or they should go to school and ask register committee to verify their file. Some rules of pre-register activity are obtained from government regulation and standard operational procedure which are accessible from PPDB website.

The activity sequence in pre – register activity start when student apply for pre – register activity. Student submit his data to system which indicates his participation in pre – register. Then student submit some information about his domicile area, student can be from the same area as the applied school, or it could be from other area. Student submit his graduation year of previous level of education (which could be in the same year as school registration or it could be on previous year) and student also submit his information about whether he graduate from official program (junior school) or from commensurate program. Student must also submit his information about his past school information (whether student graduated from local school or his past school was outside the area). The last step is system will verify all data which has been entered by student (see Fig. 8).

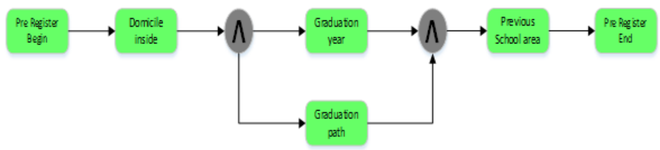

Fig. 8. Process Model of School Pre – Register Activity

E. Defining Contex Sensitive Grammar for School Registration

For study case above we build an applicable *Context Sensitive Grammar* to construct a base process. Given G_S = (N, $\sum$, R, BP), where N = (MI, MD, DA, GY, GP, SA, PRB, PRE), and $\sum$ = ((;), ∧, a_{start}, a_{end}, *dia*, *doa*, *gyt*, *gyl*, *gpj*, *gpc*, *sai*, *sao*, *prbi*, *prbc*, *prei*, *prec*). The set of production rule, R, is explained by (3) to (10).

BP → *astart; MD; aend* (3)
MD → *PRB; MI; PRE* (4)
MI → *DA; GY ∧ GP; SA* (5)
DA → *dia | doa* (6)
GY → *gyt | gyl* (7)
GP → *gpj | gpc* (8)
SA → *sai | sao* (9)

PRB;dia; gyt; gpj; sai; PRE → *prbi; dia; gyt; gpj; sai; prei* (10a)
PRB;dia; gyt; gpj; sao; PRE → *prbi; dia; gyt; gpj; sao; prei* (10b)
PRB;dia; gyt; gpc; sai; PRE → *prbc; dia; gyt; gpc; sai; prec* (10c)
PRB;dia; gyt; gpc; sao; PRE → *prbc; dia; gyt; gpc; sao; prec* (10d)
PRB;dia; gyl; gpj; sai; PRE → *prbc; dia; gyl; gpj; sai; prec* (10e)
PRB;dia; gyl; gpj; sao; PRE → *prbc; dia; gyl; gpj; sao; prec* (10f)
PRB;dia; gyl; gpc; sai; PRE → *prbc; dia; gyl; gpc; sai; prec* (10g)
PRB;dia; gyl; gpc; sao; PRE → *prbc; dia; gyl; gpc; sao; prec* (10h)
PRB;doa; gyt; gpj; sai; PRE → *prbi; doa; gyt; gpj; sai; prei* (10i)
PRB;doa; gyt; gpj; sao; PRE → *prbc; dia; gyt; gpj; sao; prec* (10j)
PRB;doa; gyt; gpc; sai; PRE → *prbc; dia; gyt; gpc; sai; prec* (10k)
PRB;doa; gyt; gpc; sao; PRE → *prbc; dia; gyt; gpc; sao; prec* (10l)
PRB;doa; gyl; gpj; sai; PRE → *prbc; dia; gyl; gpj; sai; prec* (10m)
PRB;doa; gyl; gpj; sao; PRE → *prbc; dia; gyl; gpj; sao; prec* (10n)
PRB;doa; gyl; gpc; sai; PRE → *prbc; dia; gyl; gpc; sai; prec* (10o)
PRB;doa; gyl; gpc; sao; PRE → *prbc; dia; gyl; gpc; sao; prec* (10p)

EXPLANATION

MI = *Sub-model independent*
MD = *Sum-model dependent*
DA = *Domicile area*
GY = *Previous education graduation year*
GP = *Previous education graduation path*
SA = *Previous education school area*
PRB = *Pre-register activity begin*
PRE = *Pre-register activity end*
dia = *domicile inside area*
doa = *domicile outside area*
gyt = *Previous education graduation year is this year*
gyl = *Previous education graduation year is last year*
gpj = *Previous education is junior school*
gpc = *Previous education is commensurate program B*
sai = *Previous school is inside area*
soa = *Previous school is outside area*
prbi = *Pre-register begin activity is independent*
prbc = *Pre-register begin activity is through committee*
prei = *Pre-register end activity is independent*
prec = *Pre-register end activity is through committee*

We will try to build base process from *Context Sensitive Grammar* above. The base process which will be used is basic scenario, where participant of pre-register activity has inside area of domicile location, graduate this year from junior school inside area. Derivation string for constructing base process will be shown in (11) to (19a).

BP (11)
a_{start}; *MD*; a_{end} … (12)
a_{start}; ***PRB; MI; PRE;*** a_{end} … (13)
a_{start}; *PRB*; ***DA; GY ∧ GP; SA;*** *PRE*; a_{end} … (14)
a_{start}; *PRB*; ***dia;*** *GY ∧ GP; SA; PRE*; a_{end} … (15)
a_{start}; *PRB; dia;* ***gyt*** *∧ GP; SA; PRE*; a_{end} … (16)
a_{start}; *PRB; dia; gyt ∧* ***gpj****; SA; PRE*; a_{end} … (17)
a_{start}; *PRB; dia; gyt ∧ gpj;* ***sai****; PRE*; a_{end} … (18)
a_{start}; ***prbi****; dia; gyt ∧ gpj; sai;* ***prei;*** a_{end} … (19a)

The last step of derivation string (19a) can be drawn into graph representation as in PROVOP. We add adjustment points in each node of sub-model independent as shown in Fig. 9.

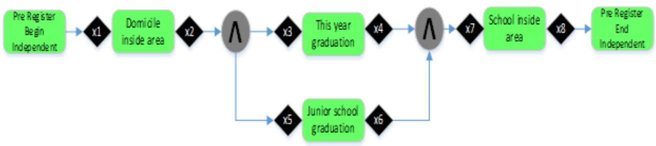

Fig. 9. Base process constructed from *Context Sensitive Grammar*

We can apply change operation in PROVOP to base process above. For first variation (variant 1), a school need to verify student who has outside area domicile and his previous school is also from outside area. Second variation (variant 2), a school need to verify student who graduates last year, and in third variation (variant 3), a school need to verify student who graduates from commensurate program B. Thus change operations needed to form those variations will be shown on Table I.

TABLE I. CHANGE OPERATIONS TO FORM VARIANT 1

Change Operation	Activity	Pre Adjustment Point	Post Adjustment Point
Delete	Domicile Outside Area	X1	X2
Insert	Domicile Inside Area	X1	X2
Delete	School Inside Area	X7	X8
Insert	School Outside Area	X7	X8

Since combination of sub-model independent schema of variant 1 is changed from base process, we have to look at production rule in *sensitive grammar.* Application of change operations resulting sub-model dependent schema to change following production rule. The production rule fulfilling variant 1 is production rule of (10j). Pre-register begin and end independent are changed into Pre-register begin and end by committee (see Fig. 10).

Fig. 10. Process model of variant 1

TABLE II. CHANGE OPERATIONS TO FORM VARIANT 2

Change Operation	Activity	Pre Adjustment Point	Post Adjustment Point
Delete	This year graduation	X3	X4
Insert	Last year graduation	X3	X4

Variant 2 (see Table II) is adapted from production rule (10e), resulting Pre-register begin and end independent are changed into Pre-register begin and end by committee (see Fig. 11).

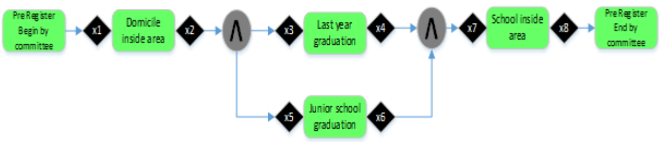

Fig. 11. Process model of variant 2

TABLE III. CHANGE OPERATIONS TO FORM VARIANT 3

Change Operation	Activity	Pre Adjustment Point	Post Adjustment Point
Delete	Junior school graduation	X5	X6
Insert	Commensurate program B	X5	X6

Variant 3 (see Table III) is adapted from production rule (10c), resulting Pre-register begin and end independent are changed into Pre-register begin and end by committee (see Fig. 12).

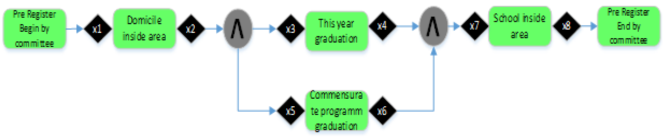

Fig. 12. Process model of variant 3

IV. CONCLUSION

We have proposed an approach which combines contextual design and context free composition. The contextual composition utilizes *Context Sensitive Grammar,* while the *Context Free Grammar* utilizes free change operations. *Context Sensitive Grammar* can be used for constructing process model properly based on right formal regulation through application of production rules. *Sensitive grammar* is used to push free change operations to fulfill ascertained rule. The result is a new flexible variant composition which matches up with production rule.

ACKNOWLEDGEMENT

We would like to thank Institut Teknologi Sepuluh Nopember and DIKTI - Directorate General of Higher Education - Ministry of Education and Culture of Indonesia for supporting the research.

REFERENCES

[1] M. La Rosa, van der Aalst W.M.P., M. Dumas, F.P. Milani: Business Process Variability Modeling: A Survey. BPM Center Report BPM-13-16, BPMcenter.org, 2013, page 2.

[2] Sarno Riyanarto, Djeni, C.A., Mukhlash,I., Sunaryono D., "Developing A Workflow Management System for Enterprise Resource Planning," *Journal of Theoretical and Applied Information Technology,* February 2015, Vol. 72, No.3, pp. 412-421.

[3] Hallerbach, A., Bauer T., Reichert, M.: Managing Process Variants in the Process Lifecycle. In: Proc. ICEIS'08 Conference. (2008), pp.154–161.

[4] Hallerbach, A., Bauer, T., Reichert, M.: Issues in Modeling Process Variants with Provop.In: Proc. of the 4th Int. Workshop on Business Process Design (BPD'08), Workshop held in conjunction with Business Process Management (BPM'08) conference. LNBIP 17, Springer (2009), pp.54–65

[5] Buntrock, G., Niemann, G.: Weak Growing Context-sensitive Grammars. Universität Würzburg, Technical Report, May 1995, page 2.

[6] Goswami, D., Krishna, K. V.: Formal Language and Automata Theory. November 2015, pp. 19-23.

[7] Dassow, J., & Masopust, T.: On Restricted Context-Free Grammar. Journal of Computer and System Sciences, 2012, pp. 293-304.

[8] Brabrand, C., Giegerich, R, Møller, A.: Analyzing Ambiguity of Context Free Grammars. In: Science of Computer Programming, Vol. 75, Issue 3, March 2010, pp. 178-180.

[9] Savage John. E.: Models of Computation: Exploring the Power of Computing. 1997, pp. 182-183.

[10] Rampersad, N., Shallit, JDetecting Patterns in Finite Regular and Context-Free language. Information Processing Letters, 2010, pp. 108-112

[11] Barash Mikhail., Okhotin, Alexander.: An Extension of Context Sensitive Grammars with One-Sided Context Specification. Information and Computation, 2014, pp. 7-8

[12] Hallerbach, A., Bauer, T., Reichert, M.: Capturing variability in business process models: the Provop approach. Journal of Software Process Improvement and Practice, 2009, pp. 7-8.