International Journal of Business and Systems Research

ISSN online: 1751-2018 - ISSN print: 1751-200X
https://www.inderscience.com/ijbsr

Reconfigurable business process: a constraint-based approach

Priyanka Chakraborty, Anirban Sarkar

DOI: 10.1504/IJBSR.2023.10046549

Article History:

Received:	07 May 2021
Accepted:	30 November 2021
Published online:	05 January 2024

Reconfigurable business process: a constraint-based approach

Priyanka Chakraborty* and Anirban Sarkar

Department of Computer Science and Engineering,
National Institute of Technology Durgapur,
West Bengal, India
Email: priyanka1nitdgp@gmail.com
Email: sarkar.anirban@gmail.com
*Corresponding author

Abstract: Continuous and rapid changes in the business environment and user requirements demand a suitable design of business processes (BPs). Most of the existing approaches either add a large number of BP elements to the existing BP or replace the current BP with a new one. Thus, those approaches are expensive and less user-friendly. Conversely, the constraint-based approach results in a minimal amount of changes in the structure of BP. Additionally, this approach gives the privilege to the stakeholders. Additionally, this approach gives the privilege to the stakeholders to reconfigure the BPs according to their goal requirements. In this paper, a constraint-driven approach is proposed to achieve reconfigurability in BP. Constraints like data, gateway and ordering are considered in BPs. Several real-life case studies have been described in detail. Finally, an algorithm named reconfigurable business process achievement algorithm (RCBPA) has been proposed and implemented to achieve reconfigurability in BP.

Keywords: constraint-based reconfigurability; gateway constraints; ordering constraints; data constraints; reconfigurable business process.

Reference to this paper should be made as follows: Chakraborty, P. and Sarkar, A. (2024) 'Reconfigurable business process: a constraint-based approach', *Int. J. Business and Systems Research*, Vol. 18, No. 1, pp.1–29.

Biographical notes: Priyanka Chakraborty is presently a Research Fellow in the National Institute of Technology Durgapur, India, since February 2016. She received her MTech in Software Engineering from the same institute in 2015, and MCA from the West Bengal University of Technology, India, in 2011. She is pursuing her PhD in the field of Software Design. Her main areas of research interests include enterprise architecture framework.

Anirban Sarkar is presently an Associate Professor in the Department of Computer Science and Engineering, National Institute of Technology Durgapur, India. He received his PhD from the National Institute of Technology Durgapur, India, in 2010. His broad areas of research interests are software engineering, database management system and cloud computing. He is an author or co-author of over 100 publications in numerous refereed journals and conference proceedings.

1 Introduction

Business processes (BPs) are now seen as business services (BSs), which extend the organisational boundaries and need to satisfy the cross-organisational objectives. In this context, the BP can be defined as "a business process is a collection of activities that takes one or more kinds of input and creates an output that is of value to the customer" (Monk and Wagner, 2012). Technically, BPs are realised as service oriented architecture (SOA) which is a service-based software integration solution for related business activities (Li et al., 2005; Basias et al., 2013; Sarno et al., 2015). Reusability, flexibility and maintainability are the main powers of service-oriented architecture (Ardagna and Pernici, 2007).

A BP is comprised of five types of elements including, a set of activities or user tasks, a set of sequence flows, a set of gateways, a set of events and data or messages. Those five BP elements and the BP flow are immensely affected by the three major constraints like data, gateway and ordering constraints. Moreover, the three major BP elements such as data, gateway and activity and their corresponding constraints like data, gateway and ordering constraints determine the execution path of the BP based on the various conditions. Those three constraints are general and mandatory, however, there may exist specific application-based specific constraints like event-based constraints, role constraints to affect the BP flow to a certain limit. Activities or user tasks are considered as the atomic units of a BP. The same activities can be reused in different BPs. An activity can be related to another activity, gateways, or events through sequence flows. The rearrangement, elimination, or addition of those five types of BP elements makes a BP reconfigurable so that it can adapt to the new business goals or sub-goals to a certain pre-defined limit. Reconfigurability of BPs can be achieved at either of the external or internal levels depending on the changes in the external or internal business scenarios of an organisation (Arsanjani and Ng, 2002; Bazoun et al., 2014; Marques et al., 2017). Internal level reconfigurability of a BP is the topic of concern in this paper.

The rapid and unpredictable changes in customer requirements and marketing strategies of a business organisation demand introduction of reconfigurable BPs to adapt to the ever-changing business environment (Gao and Miao, 2013; Silva et al., 2016; Chakraborty and Sarkar, 2017, 2019). Reconfigurability property enables those design capabilities in BPs by refining the overall business goals and behaviour patterns of the resultant services in an enterprise computing environment. Reconfigurable BPs are reusable, flexible and cost-effective. Moreover, those can quickly respond to the market and business environment changes. Suppose, a company aims to sell only dresses for the children, later it decides to open a new segment to sell dresses also for the adults. Here, the goal (selling dresses) remains the same but a new sub-goal has been added with the previous set of sub-goals. There is no need to replace the associated BPs to satisfy the newly added sub-goal. The new sub-goal can be sufficed by only the reconfigurable BPs. Thus, new BPs are capable of satisfying both the previous and new sub-goal simultaneously. In this context, the major research questions are:

1 How BPs are made reconfigurable?

2 At which range (maximum range) a BP is reconfigurable.

Here, the maximum range signifies the highest range of retaining the same existing services along with the new services that have been integrated into the new BP. The aim

is to make the best use of the pre-existing services by reusing them in the newly modified BP and to reduce the number of new services (Li et al., 2011; Mendonça et al., 2013; Yin et al., 2017) to be integrated into that new BP. If the range of reconfigurability of a BP exceeds that maximum value, then that BP is not reconfigurable, and a new BP is to be created.

Three major ways to achieve reconfigurability in BPs are:

1 Goal has been changed from the previous one. The newly modified goal has been availed by the rearrangement of several BP elements within a BP. In this scenario, no new BP elements have been added.

2 Goal has been reformed, and the new goal is achieved either by adding a large number of BP elements (user tasks, gateways and events) to the existing BP or by replacing the previous BP with a new BP.

3 Previous goal is transformed into a new goal through minimum changes of its sub-goals. Consequently, the new sub-goal is achieved with nominal changes of the elements of the BP, associated with that sub-goal.

The first approach is applicable only when the goal is changed to its lowest amount by merely the rearrangement of the elements in the BP, associated with that goal. In reality, this approach is not very feasible because the ever-changing business environment requires a certain amount of changes in its goal and corresponding sub-goals and those changes cannot be sufficed by only rearranging several BP elements. In the second approach, a large amount of BP elements or the replacement of the previous BP with the new BP results in the maximisation of both the development and the maintenance cost. These costs can be reduced by following the third approach which states that the incorporation of a minimal amount of BP elements into the pre-existing BP is sufficient to satisfy the modified goal. Consequently, the third approach is more feasible, cost-effective, easy to implement and less time-consuming in comparison with the other two approaches.

Most of the existing works followed either the first or the second approach, whereas the third approach is adopted in this paper. Consideration of the constraint property of the BP helps to incorporate the minimum number of new elements in the pre-existing BP to suffice the newly modified goal. Moreover, constraint specification plays an important role in specifying the boundary or limit of the domain of BPs. By changing only, the constraint specifications, the BP can be made reconfigurable. Consequently, constraint-based reconfigurable BPs (Gao et al., 2006; Li et al., 2011) become more user-centric.

In this paper, constraint-based reconfigurable BP has been introduced. Whenever the operating environment and business requirements are changed, reconfigurable BPs change their behaviour accordingly, to adapt to such changes. Usually, either the prior BP is replaced by a new BP, or an enormous number of new BP elements are incorporated into the prior set of BP elements within the BPs. Thus, both approaches (replacement of BP and the integration of new BP elements to the existing BP) cause an increase in the development and maintainable cost of BPs. In contrast, rearrangement of the existing BP elements within a BP or incorporation of a minimal number of BP elements within an existing BP is a cost-effective procedure to achieve reconfigurability in BPs.

Constraint-based reconfiguration makes BPs reusable, more flexible, more user-centric and cost-effective. The constraints are divided into three categories like:

1 data constraints

2 ordering constraints

3 gateway constraints.

Formal representations of all these three constraints are presented in this paper. Business process model and notation (BPMN) (Chinos and Trombetta, 2009; Kurz, 2016) have been used to represent the constraints pictorially. Finally, an algorithm has been proposed and implemented to achieve reconfigurability in the BP.

This paper is structured with the following sections: Section 2 encompasses the related work. Section 3 contains the representation of behavioural aspects of a BP. Section 4 is comprised of the proposed work and the proposed algorithm with its implementation. Section 5 contains one case study for the illustration of the algorithm through the representation of equivalent BP diagrams related to each constraint. Section 6 contains experimental results and their analysis. Section 7 contains the conclusions.

2 Related work

Earlier existing works in this field followed mainly the second approach to achieve reconfigurability in a BP (Yu and Lin, 2005; Hermosillo et al., 2010). This approach leads to either the addition of lots of BP elements to existing BP or the replacement of the existing BP by a refined and new BP. Consequently, that method is expensive and less user-friendly. In Zhang et al. (2012), a proactive service selection and reconfiguration approach has been proposed to prevent service process failure. Both dependent and independent context models are introduced to describe the context model transition relationship. A probabilistic graph model is used to represent the transition relationship among context elements.

The independent context model can be easily adopted, while the dependent model contains dependency between control states. Finally, context transition prediction algorithms for both context models are represented. In Zhang et al. (2005), a workplace design framework has been proposed. This framework automatically analyses the BPs and produces workplace applications from those BPs. In Xiao et al. (2011), a constraint-based framework has been proposed. It can dynamically adapt the BPs. Process fragments are modular and reusable functional units. Depending on the process fragments, process adaptation can be in two levels – the process schema level and the fragment level. At the process schema level, the fragments and their relationships are modelled by constraints. Whenever business requirements change a new process, schemas are dynamically generated depending on the constraints. The fragment level focuses on fragment selection and substitution. Finally, an algorithm for process generation has been proposed. In Jabbar et al. (2015), a BP reconfiguration method based on UML and polychromatic sets (PS) theory have been proposed. Finally, a BP reconfiguration for enterprise information systems has been suggested.

3 Representation of functional constraints in BP

Sometimes, pre-existing BPs may be outdated, and they should be replaced by new BPs. Moreover, customer needs, business rules and policies, operating environment in which BPs operate change abruptly and in an undefined manner from time to time. Thus, reconfigurable BPs are required to cope with the ever-changing operating environment and business requirements easily.

BPs are composed of a set of BP elements, like, activities or user tasks, gateways, events, sequence flows and data. Same BP elements can be reused across multiple BPs. Suppose, goal G_A has been modified into goal G_B where G_B is the superset of G_A. Corresponding BP (BP_A) of G_A will be also modified into the BP (BP_B) associated with G_B, so that BP_B can be the superset of BP_A that is $BP_A \subseteq BP_B$. Hence, BP_B contains several additional BP elements with the existing BP elements of BP_A.

In this work, the relationship among all the elements is specified in a constraint-based manner. Constraint-based reconfiguration of the process makes BPs reusable, more flexible, user-centric and cost-effective. It makes BPs adapt to the ever-changing internal and external business environment easily. Constraint-based reconfigurability can be achieved through three techniques like changing the range of data, adding some constraints in gateway level and changing the order of the activities.

A BP is comprised of five types of elements like the set of activities ($\overline{Ac}$) or user tasks, set of events ($\overline{Ev}$), set of gateways ($\overline{Gw}$), set of data ($\overline{Da}$) and set of sequence flows ($\overline{Sq}$). The set of constraints is considered as the sixth kind of BP element. Thus, BP can be denoted as $BP \rightarrow (\overline{Ac} \cup \overline{Ev} \cup \overline{Gw} \cup \overline{Da} \cup \overline{Sq} \cup \overline{Co})$. Activities, events, gateways, data or messages are structural elements and constraints are behavioural elements of a BP.

BP activities are of two types, like, set of tasks (*Ta*) (atomic activities) and a set of sub-processes (*SP*) (composite activities). Both types of activities are reusable through different BPs. Events may be of five types like start events (*SEv*), intermediate events (*IEv*), timer events (*TEv*), message events (*MEv*) and end events (*EEv*). Gateways are of five kinds like a parallel gateway (*PGa*), inclusive gateway (*IGa*), exclusive gateway (*XGa*), event gateway (*EGa*) and complex gateways (*CGa*).

Specification of constraints of different levels can be of three types, data level constraint specification (*DC*), gateway level constraint specification (*GC*) and ordering level constraint specifications (*OC*). Formal representation of the structural and behavioural elements within a BP can be represented as,

$$\forall x\big(Ele(x) \wedge Type(x)\big) \rightarrow Ac(x) \vee Ev(x) \vee Gw(x) \vee Da(x) \vee Sq(x) \vee Co(x) \quad \text{(F1)}$$

$$\forall x\big(Ac(x) \wedge Type(x)\big) \rightarrow Ta(x) \vee Sp(x) \quad \text{(F2)}$$

$$\forall x\big(Gw(x) \wedge Type(x)\big) \rightarrow PGa(x) \vee IGa(x) \vee XGa(x) \vee EGa(x) \vee CGa(x) \quad \text{(F3)}$$

$$\forall x\big(Ev(x) \wedge Type(x)\big) \rightarrow SEv(x) \vee IEc(x) \vee TEv(x) \vee MEv(x) \vee EEv(x) \quad \text{(F4)}$$

$$\forall x\big(Co(x) \wedge Type(x)\big) \rightarrow \big(DC(x) \vee GC(x) \vee OC(x)\big) \quad \text{(F5)}$$

$$\begin{aligned}&\forall y1\forall y2\exists t1\exists t2\big(Ac(y1)\wedge Ac(y2)\wedge timestamp(t1)\wedge timestamp(t2)\\&\wedge Execute(y1,t1)\wedge Execute(y2,t2)\wedge Greater\_than\_equal\_to(t1,t2)\big)\\&\rightarrow\exists t3\exists t4 timestamp(t3)\wedge timestamp(t4)\wedge Execute(y1,t3)\wedge Execute(y2,t4)\\&\wedge Greater\_than\_equal\_to(t3,t4)\end{aligned}\tag{F6}$$

In F1, *Ele*() predicate implies the element of a BP and *Type*() represents the type of its argument. *Ac*(), *Ev*(), *Gw*(), *Da*(), *Sq*() and *Co*() predicates imply activities, events, gateways, data, sequence flows and constraints elements of a BP, respectively. In F2, predicates *Ta*() and *Sp*() imply two types of activities like a task (atomic activity) and sub-process (composite activity), respectively. In F3, predicate *PGa*() indicates parallel gateway or AND gateway, *IGa*() represents Inclusive gateway, *XGa*() implies exclusive or EX-Or gateway, *EGa*() represents event gateway and *CGa*() implies complex gateways. In F4, predicates, *SEv*(), *IEv*(), *TEv*(), *MEv*() and *EEv*() implies start event, intermediate event, timer event, message event and end events, respectively. In F5, predicates *DC*(), *GC*() and *OC*() represent data constraint, gateway constraints and ordering constraints, respectively. In F6, the *timestamp*() predicate denotes the time and $t1$, $t2$ are instances of timestamp. Similarly, $y1$ and $y2$ are instances of activities. *Execute*() predicate implies that $y1$ and $y2$ instances are executed at timestamp $t1$ and $t2$ respectively. *Greater_than_equal_to*() predicate implies that the second argument is greater than or equal to the value of the first argument. Thus, F6 implies that a BP will be reconfigurable and maintain the temporal aspect property if the activities of the BP maintain a specific order of execution in the previous and the new modified BP.

The detailed formalism of all those three constraints can be expressed as follows.

3.1 Data or message constraints (DC)

The BP elements like activities, gateways and events may receive data as input and generate the output data. However, the data flow may be a constraint in different ways. It can be of three types, data type constraints (*DTC*), not null constraints (*DNLC*) and *value range constraints* (*DVRC*). *DTC* defines what should be the type of input dataset and output dataset. It also specifies that the input data type and output data type must be compatible. *DNLC* specifies that the value of the data must not be null. *DVRC* describes that the value range of the data domain must lie between the maximum and minimum values specified by the user.

$$\forall x\big(DC(x)\wedge Type(x)\rightarrow DTC(x)\wedge DNLC(x)\wedge DVRX(x)\big)\tag{F7}$$

$$\begin{aligned}&\forall x\big(DTC(x)\rightarrow\exists y\exists z\big(Ac(y)\vee Gw(y)\vee Ev(y)\big)\wedge Input(x,y)\wedge Output(z,y)\\&\wedge EQ\big(DT(x),DT(z)\big)\big)\end{aligned}\tag{F8}$$

$$\forall x\Big(DNLC(x)\rightarrow\exists y\big(Ac(y)\wedge Input(x,y)\wedge Not\_Equal\_To\big(value(x),0\big)\big)\Big)\tag{F9}$$

$$\begin{aligned}&\forall x\big(DVRC(x)\big)\rightarrow\exists y\big(Gw(y)\wedge\wedge Input(x,y)\\&\wedge Greater\_Than\_Equal\_To\big(value(x),\mathrm{Min}\big)\\&\wedge Less\_Than\_Equal\_To\big(value(x),\mathrm{Max}\big)\big)\end{aligned}\tag{F10}$$

In F7, *DC*() predicate represents the data constraint. *DTC*(), *DNLC*() and *DVRC*() represent three types of data constraints like data type constraints, not null constraints and value range constraints, respectively. In F8, predicate *Input*() implies that the first argument is input to the second argument. Similarly, the *Output*() predicate denotes that the first argument is the output of the second argument. *DT*() function implies the data type of its argument. *EQ*() predicate implies that its two arguments are equal in value. In F9, the *value*() function returns the value of its argument and the *Not_Equal_To*() predicate implies that the value of its first argument is not equal to zero. In F10, *Greater_Than_Equal_To*() predicate denotes that the value of the first argument is greater than or equal to the Min(constant value) value. *Less_Than_Equal_To*() predicate implies that the value of the first argument is less than or equal to the Max (constant value) value. The Min and Max values are set by the user.

3.2 Gateway level constraints (GC)

In a BP, gateways control the flow of the process. Gateways are of five types, namely:

1 exclusive gateway (*XGa*)

2 inclusive gateway (*IGa*)

3 parallel gateway (*PGa*)

4 event-based gateway (*EGa*)

5 complex gateway (*CGa*).

These types of constraints restrict the flow of the elements within a BP. Exclusive gateway evaluates the state of a BP depending on various conditions and divides the flows into two or more paths. Only one of those paths will be active at a time, i.e., all the paths are mutually exclusive. An inclusive gateway breaks the process flow into two or more paths. Unlike, exclusive gateway, an inclusive gateway can choose more than one path at a time. Parallel gateways are used to demonstrate the simultaneous occurrences of two tasks. Event gateway works like an exclusive gateway, but it checks what event has happened. Depending on an event, this gateway divides the process flow into more than one path. Only one path can be activated at a time.

$$\forall x1 \forall x2 \left(Ac(x1) \wedge Ac(x2) \wedge XGa(x1, x2) \right) \rightarrow (\neg x1 \wedge x2) \vee (x1 \wedge \neg x2) \quad \text{(F11)}$$

$$\forall x1 \forall x2 \left(Ac(x1) \wedge Ac(x2) \wedge IGa(x1, x2) \right) \rightarrow (x1 \vee x2) \quad \text{(F12)}$$

$$\forall x1 \forall x2 \left(Ac(x1) \wedge Ac(x2) \wedge PGa(x1, x2) \right) \rightarrow (x1 \wedge x2) \quad \text{(F13)}$$

$$\forall x1 \forall x2 \left(Ev(x1) \wedge Ev(x2) \wedge EGa(x1, x2) \right) \rightarrow \left((\neg x1 \wedge x2) \vee (x1 \wedge \neg x2) \right) \quad \text{(F14)}$$

In F11, F12, F13 and F14, predicates *XGa*(), *IGw*(), *PGa*() and *EGa*() imply exclusive or EX-Or gateway, inclusive gateway, parallel gateway or AND gateway and event gateway, respectively.

3.3 Ordering constraints (OC)

All activities within a BP are performed either in a particular order or in non-sequential order. The order of execution of activities must be preserved properly. A small change in the execution order of activities results in a modified BP.

$$\forall x1 \forall x2 \exists t1 \exists t2 \big(Ac(x1) \wedge Ac(x2) \wedge Seq\_order(x1, x2) \wedge timestampl(t1) \\ \wedge timestampl(t2) \big) \rightarrow (x1 \wedge x2) \wedge Execute(x1, t1) \wedge Execute(x2, t2) \\ \wedge Greater\_than(t1, t2) \tag{F15}$$

In F15, *Seq_order*() predicate implies the sequential order of its two arguments. The *timestamp*() predicate implies time and $t1$, $t2$ are instances of timestamp. *Execute*() predicate implies that the first argument is executed at the timestamp of the second argument. *Greater_than*() predicate implies that the second argument is greater than the first argument.

3.4 Priority of constraints

Gateway constraints influence the flow of a BP at the most as the inclusion of a gateway determines the path of execution of the activities within a BP. If gateway constraint exists then only, data constraints and ordering constraints come into existence. Data constraints affect the subsequent activities, whereas, ordering constraints affect the pre-condition and the post-condition of the sequence flows within a BP. When there is a requirement of modification in more than one constraint, then priority values are needed to be assigned to those constraints so that the execution order of the BP elements can be structured according to those priority values of the constraints. The gateway constraint is given the highest priority due to its maximum impact on the BP flow while the ordering and the data constraints are given lesser priority values because of their lower effect on BP flows. Thus, the priority values of various constraints turn a BP into a reconfigurable BP in a more user-defined manner. Moreover, whenever *GC*, *DC* and *OC* exist simultaneously *GC* will be executed first followed by *DC* and *OC*. It can be denoted as $GC \rightarrow DC \wedge OC$.

4 Proposed reconfigurability mechanism in BP

The procedure for achieving constraint-based reconfigurable BP is comprised of multi-fold steps. The first step is to identify the sub-goals to be changed. Then, the BPs, associated with those sub-goals will be recognised. The following step is to identify the BP components that are required to be reconciled. Then, data present in the same activities within the same process will be identified and merged.

4.1 Method

Any BP is deployed to achieve a set of business goals. Each goal may be comprised of several sub-goals. Whenever any of these sub-goals is changed, or a new sub-goal is added, the associated BP is also changed. New BP (BP') satisfies both the former and the new sub-goal. Thus, BP' is the superset of the previous BP ($BP \subseteq BP'$).

The entire procedure for achieving reconfigurable BP includes different steps. At *first*, the sub-goal is changed. Either a new sub-goal is added, or an existing sub-goal is to be modified. *Secondly*, identify the associated BP that is to be modified to satisfy the changed or newly added sub-goal.

The next step is to identify which constraint level modification is required. It may be either at the data constraint level modification, at the gateway constraint level modification or the ordering level constraint modification. It may also be any combination set of those three constraints. Whenever more than one constraint level change is required, the execution order of the constraints is maintained by the priority values related to each of them. Priorities are determined by the users depending on the input and output sequence flows of the BP elements. Gateway level constraint modification has been given the highest priority. All the constraints to be changed are stored in a priority queue with their priority values. While the priority queue is non-empty, the constraint with the highest priority is fetched from the queue, and corresponding activities are executed.

The change in the data level constraint is defined by the change in the range value of the data domain, depending on the related condition specified by the stakeholders. In gateway level constraint changes, a new type of gateway (exclusive, inclusive, parallel, event and complex) replaces the old one. In ordering level constraint specification, the order of the execution of a set of activities is rearranged based upon the current business requirements.

All the constraints are stored in a priority queue based upon their priority values. Priority values of the constraints are assigned according to their order of execution. At first, the input dataset is classified into distinct classes, depending on the value range of the data constraint. After categorisation, different classes of data are given as input into different types of gateways based on their related checking conditions.

Different execution orders of a set of activities result in a distinct set of outputs. Thus, in the ordering constraint, the execution order of the set of activities keeps changing. Data constraints and ordering constraints are given the same priority values. Thus, whenever more than one constraint level is needed to be changed in the priority queue; they are executed according to their priority values.

4.2 Metrics

In this section, three software metrics, namely:

1 degree of reusability (D_{RU})

2 degree of reappropriation (D_{RA})

3 degree of reconfigurability (D_{RC}) are introduced.

D_{RU} is used to measure how many elements of the previous BP are reusable in newly modified BP. D_{RA} is represented to calculate the reappropriation, required to carry out in each of those BP elements, obtained in D_{RU}. D_{RC} is formulated to estimate how much complexity exists to change each type of BP element in terms of gateway (Ga), activity (Ac) and data (Da) type by assigning certain weightage values to Ga, Ac and Da.

a Degree of reusability (D_{RU}) is represented as the ratio [equation (1)] of the number of reusable BP elements (*NRU*) from the pre-existing BP (*BP*1) and the total number of BP elements (*NT*) in the resultant BP (*BP*′).

$$D_{RU} = \frac{NRU}{NT} \tag{1}$$

$D_{RU} \leq k$, where $k \leq 1$ and k are user-defined. If ($D_{RU} \leq k$) condition is satisfied then only D_{RA} and D_{RC} are calculated. Finally, in this step, a vector containing all the elements of the previous BP, those will be reused in the new BP (*BP*′), is generated.

b For each of the BP elements belonging to the vector obtained in the previous step, D_{RA} is calculated. Thus, the number of changes that are required for three types of BP elements (*Da*, *Ga* and *Ac*) are calculated. Let us take an example. Suppose, a gateway is changed by adding a new branch to it as well as the type of that *Ga* is also changed like the parallel (AND) *Ga* is turned into an exclusive (X-OR) *Ga*. Consequently, changes in two features of that *Ga* result in changes in the level is average. Similarly, if a single feature of a BP element is changed then the changes in level are simple and changes in three features consequence changes in level are complex. Levels complex, average and simple are also termed as high (*Hi*), medium (*Md*) and low (*Lw*), respectively. So, it is clearly understood that change in level simple is the most desirable and the changes in the level complex are the least desirable. Now, certain weightage values are assigned to three types of BP elements like *Ga*, *Ac* and *Da* as well as to the levels *Hi*, *Md* and *Lw* for each of those three elements individually based on their modification effect on the entire BP. Any kind of change in the gateway affects the BP most while the effects of changes in the activity and the data within a BP are medium and minimum, respectively. Thus, $[\{Wv(Da(Lw)) < Wv(Da(Md)) < Wv(Da(Hi))\} < \{Wv(Ac(Lw)) < Wv(Ac(Md)) < Wv(Ac(Hi))\} < \{Wv(Ga(Lw)) < Wv(Ga(Md)) < Wv(Ga(Hi))\}]$, where $Wv(Ga(Lw))$ denotes the weightage value of the gateways those belongs to the *Lw* level modifications.

$\Rightarrow [\{x_1 < x_2 < x_3\} < \{y_1 < y_2 < y_3\} < \{z_1 < z_2 < z_3\}]$, where $x_1, x_2, x_3, y_1, y_2, y_3, z_1, z_2, z_3$ are user-defined variables those values are decided by the stakeholders of an EAF based on the current business scenario. Thus, in this step of calculation D_{RA}, a 3D vector is obtained containing three elements, each of which represents the sum of weightage values of data elements, activity elements and gateway elements, respectively.

$$D_{RA} = \begin{bmatrix} Wv(Da(Lw)) + Wv(Da(Md)) + Wv(Da(Hi)) \\ Wv(Ac(Lw)) + Wv(Ac(Md)) + Wv(Ac(Hi)) \\ Wv(Ga(Lw)) + Wv(Ga(Md)) + Wv(Ga(Hi)) \end{bmatrix} = \begin{bmatrix} (x1 + x2 + x3) \\ (y1 + y2 + y3) \\ (z1 + z2 + z3) \end{bmatrix} \tag{2}$$

In equation (2), the first element is the summation of $Wv(Da(Lw))$, $Wv(Da(Md))$, and $Wv(Da(Hi))$, that denotes the total weightage value of data of level *Lw*(simple), *Md*(average) and *Hi*(complex). Similarly, the second element is the summation of $Wv(Ac(Lw))$, $Wv(Ac(Md))$, and $Wv(Ac(Hi))$ that signifies the total weightage value of activities of level *Lw*(simple), *Md*(average) and *Hi*(complex) and the third element is

the summation of *Wv*(*Ga*(*Lw*)), *Wv*(*Ga*(*Md*)), and *Wv*(*Ga*(*Hi*)), that denotes the total weightage value of gateway of level *Lw*(simple), *Md*(average) and *Hi*(complex).

c Degree of reconfigurability (D_{RC}) is represented as the ratio of the magnitude of two vectors. The column matrix (3 ∗ 1) is comprised of three rows (*Da*, *Ac*, *Ga*) and one column named as Max_weightage value (*MWv*).

Now, another calculation is required for computing how many new BP elements are needed to be created to suffice the new sub-goal. Consequently, the newly created BP (*BP*2) associated with the new sub-goal will be created and finally pre-existing BP (*BP*1), related to the previous sub-goal will be merged with the newly created BP (*BP*2) to suffice the newly modified goal. Thus, the new BP (*BP′*), produced as the output, is the combined form of the *BP*1 and *BP*2. So, another 3D weightage vector with three elements is generated. The first element represents the total weightage value of data elements belonging to the levels such as *Lw*(*d*1), *Md*(*d*2) and *Hi*(*d*3) while the second element denotes the total weightage value of activity elements belonging to the levels like *Lw*(*a*1), *Md*(*a*2) and *Hi*(*a*3). Similarly, the third element of the 3D vector symbolises the total weightage value of activity elements belonging to the levels like *Lw*(*g*1), *Md*(*g*2) and *Hi*(*g*3). And rows named as *Da*, *Ac* and *Ga* and one column entitled as *Wv*, is generated for measuring how many elements from *BP*2 are included in *BP′*.

Thus, the 3D weightage vector is obtained as

$$WM = \begin{bmatrix} (d1 + d2 + d3) \\ (a1 + a2 + a3) \\ (g1 + g2 + g3) \end{bmatrix} \tag{3}$$

Finally, D_{RA} and *WM* are added by the rule of 3D vector addition and the result is like,

$$TW = \begin{bmatrix} ((x1 + x2 + x3) + (d1 + d2 + d3)) \\ ((y1 + y2 + y3) + (a1 + a2 + a3)) \\ ((z1 + z2 + z3) + (g1 + g2 + g3)) \end{bmatrix} \tag{4}$$

where, three elements of the *TW* vector represent the total weightage values of data, activities and gateways that either have been taken from *BP*1 and reused in *BP′* with a certain level of modification or newly created in *BP*2 and is reused in *BP′*. A new vector denoted as *NTW* is introduced to represent the total weightage values of data, activities and gateways, which needed to be incorporated within *BP′* if *BP′* would be newly created to accomplish the newly modified goal *G′*.

$$NTW = \begin{bmatrix} Wvn(Da(Hi)) \\ Wvn(Ac(Hi)) \\ Wvn(Ga(Hi)) \end{bmatrix} \tag{5}$$

where *Wvn*(*Da*(*Hi*)) represents the total weightage value of data elements, contained in *BP′*, if *BP′* would be newly created in its totality, without reusing any of its elements.

Thus, D_{RC} is represented as the ratio of the magnitude of two vectors like *TW* and *NTW*.

$$D_{RC} = \frac{|TW|}{|NTW|} \leq x < 1 \tag{6}$$

where the value of the variable x is decided by the stakeholders of an enterprise based on the present business scenario as well as the ongoing customer requirements and that value must be less than 1. The value of D_{RC} increases with the higher values of TW, that means the amount of changes in the pre-existing elements from $BP1$ and $BP2$, those are integrated within BP' is of a large amount. Consequently, the cost for this reconfiguration is also high. On the contrary, a lower value of D_{RC} implies that a small amount of changes in the prior BP elements in $BP1$ and $BP2$ is sufficient for the incorporation of those BP elements within BP' to satisfy the modified goal G'. Therefore, the cost of reconfiguration is also reduced to a certain extent. So, minimising the value of D_{RC} is the prime focus of the stakeholders at the time of achieving the newly modified goal G'. Thus, the user-defined value of D_{RC} gives the privilege to its users to take favourable decisions based on the current market situation, to earn the maximum profit for their organisation.

4.2.1 Illustration of metrics with example

Let us explain the above three software metrics with the help of an example. Suppose, a BP' is comprised of 21 BP elements (10 *Da*, 7 *Ac*, 4 *Ga*) and 14 elements have been reused from pre-existing BP ($BP1$). Among those 14 elements from $BP1$, 11 elements can be reused in their original form and the rest of the elements are reused in BP' through further modifications.

So, $D_{Ru} = \frac{14}{21} = 0.66 \geq k$, where k is determined as 0.3 by the stakeholders.

The value of k conceptually may vary from 0 to 1. The lowest value (0) of k signifies that no element, belonging to the prior BP ($BP1$) can be reused in the BP'. Consequently, BP' would be fully reconfigured and it would increase the development cost to the maximum level. Hence, in that case, it would be profitable to reconstruct BP' instead of reconfigure it. On the other hand, the highest value (1) of k denotes that all BP elements of $BP1$ can be reused in BP' and that is the ideal condition. In reality, the suitable value of k is assigned by the top-level stakeholders like the management group of an organisation, based on their various types of requirements, business strategies and the margin of profit level.

So, the $BP1$ is reusable. A list is obtained containing those elements to be reused like, [*Ac*1, *Ac*3, *Ac*4, *Ac*5, *Ac*7, *Gw*1, *Gw*2, *Gw*4, *Da*1, *Da*2, *Da*4, *Da*5, *Da*6, *Da*8]. Suppose, stakeholders has assigned the values of x_1, x_2, x_3, y_1, y_2, y_3, z_1, z_2, and z_3 as 1, 2, 3, 4, 5, 6, 7, 8 and 9, respectively.

The activities like, *Ac*1 and *Ac*3 are to be changed in terms of one feature (level *Lw*) and two features (level *Md*) respectively while only one feature (level *Lw*) is to be changed in *Gw*1. Other BP elements can be reused with their original form. So, we obtain a 3D vector $D_{RA} = \begin{bmatrix} (0+0+0) \\ (4+5+0) \\ (7+0+0) \end{bmatrix} = \begin{bmatrix} 0 \\ 9 \\ 7 \end{bmatrix}$.

Suppose, four *Da*, two *Ac* and one *Ga* are taken from *BP*2 with the complex (*Hi*) level of changes and those are reused in *BP′*. So, the $WM = \begin{bmatrix} 0+0+(4*3) \\ 0+0+(2*6) \\ 0+0+(1*9) \end{bmatrix} = \begin{bmatrix} 12 \\ 12 \\ 9 \end{bmatrix}$.

So, the desired 3D vector *TW* is obtained after the addition of D_{RA} and *WM* $\begin{bmatrix} 0 \\ 9 \\ 7 \end{bmatrix} + \begin{bmatrix} 12 \\ 12 \\ 9 \end{bmatrix} = \begin{bmatrix} 12 \\ 21 \\ 16 \end{bmatrix}$, The magnitude of the vector *TW* is $\sqrt{12^2 + 21^2 + 16^2} = 29.$

If *BP′* would be newly created in its wholeness then the weightage vector *NTW* would be $\begin{bmatrix} (10*3) \\ (7*6) \\ (4*9) \end{bmatrix} = \begin{bmatrix} (30) \\ (42) \\ (36) \end{bmatrix} = \sqrt{30^2 + 42^2 + 36^2} = 62.928.$

$$D_{RC} = \frac{29}{62.928} = .460 \leq x = .75$$

Hence, the BP is reconfigurable.

If $0.5 \leq D_{RC} < 1$ then only BPs are reconfigurable. The boundary value of D_{RC} is specified by the user. Thus, the user can change the limiting value of D_{RC} based on the current scenario of the enterprise.

4.3 Algorithm for reconfigurable BP

Currently, various types of organisations are inefficient to deal with the fast and irregular changes in business strategies, customer requirements and behaviour. These rapid and unpredictable changes in internal and external business environments demand business organisations to adopt reconfigurable BPs to efficiently handle this random nature of business organisations.

An algorithm has been proposed in this section to achieve reconfigurability in BP. When a goal or a sub-goal has been changed, the related BP is also changed. Thus, corresponding BP elements to be changed are recognised first. The degree of reconfigurability D_{RC} metric has been introduced to measure the amount of changes in BP. Then, the constraint level to which the modification to happen is identified. Modifications may occur either of data constraint level, gateway level, or ordering level.

4.3.1 Discussion

The RCPBA algorithm starts with identifying the sub-goal to be changed. Consequently, the associated BP is to be modified. Then, D_{RC} is calculated for that BP. The array *E*[] stores the BP elements to be changed. *Parse*() function is defined on the array *E*[] for traversing all elements. Three functions like, *Checkactivity*(), *Check_gateway*() and *Check_flow_order*() are defined on the array *E*[]. If ($D_{RC} < 1$) for a BP, then that BP is reconfigurable. If BPs are reconfigurable, then corresponding constraint names (to be modified) are stored in an array *X*[]. Three invariants like data_constraint, ordering_constraint and gateway_constraint take inputs of the pre-existing BP (*BP*1) and the new BP (*BP*2) associated with the new sub-goal and produces the output as the new

reconfigured BP (*BP′*) by using the rule of data constraint, ordering constraint and gateway constraint, respectively. In our study, the gateway constraint has been given the highest priority as the BP flow is determined by the gateways. Data and ordering constraints have been given the same priority. So, if the array *X*[] contains gateway then a priority queue *Q* is defined where the constraint names and corresponding priorities are stored. A lower number represents a higher priority. Three operations, *Insert*(), *Get_highest_priority*(), *Delete_highest_priority*() are defined on the priority queue. All the operations can be represented using a heap data structure. The constraint with the highest priority can be accessed from the priority queue using the mean heap property (minimum the value, higher the priority). *Delete_highest_priority*() function deletes the constraint with the highest priority from the priority queue. After reconfiguring BPs corresponding to the highest priority constraint in *Q*, that constraint is removed from the *Q*. This process is continued until the *Q* becomes empty. If there is no gateway constraint in *X*[], then the priority queue is not used as the data and ordering constraints have been given the same priorities. In this case, BPs corresponding to data and ordering constraints are reconfigured in first come first serve (FCFS) manner. Finally, the entire process is terminated if *X*[] becomes empty.

Algorithm 1 Reconfigurable business process achievement algorithm (RCBPA)

Input: previous goal *G*, previous business process BP, new goal *G′*;

Output: new business process BP′;

Define two empty arrays *E*[] and *X*[] ;

Define a priority queue *Q* to store the constraints to be modified;

Define a string variable *Chp* to store the constraint name;

1 Identify the sub-goal to be changed;

2 *Y* ← the changed sub-goal;

3 Identify the name and number of BP elements to be changed;

5 *E*[] ← the BP elements to be changed;

5 Calculate degree of reconfigurability D_{RC} with the help of *E*[];

6 If ($D_{RC} < 1$)

```
{
    Print 'all BPs are reconfigurable';
    X[] ← the constraints to be modified;
    If (X[] contains 'gateway')
    {
        Q ← store the constraint names and their corresponding priorities;
        While (Q is not empty)
        {
            Chp ← retrieve the highest priority constraint name from Q;
            If (Chp == 'data')
            {
            Reconfigure BPs using data_constraint;
            Remove the head of the priority queue Q;
            }
```

```
        Else If (Chp == 'order')
                {
                Reconfigure BPs using ordering_constraint;
                                Remove the head of the priority queue Q;
        }
        Else If (Chp == 'gateway')
                {
                Reconfigure BPs using gateway_constraint;
                                Remove the head of the priority queue Q;
                }
                Else
                        Print 'wrong constraint input';
        }
    else
    {
        While(X[] is not empty)
        {
                If (X[index] == 'data')
                {
                Reconfigure BPs using data_constraint;
                }
                Else If (X[index] == 'order')
                {
                Reconfigure BPs using ordering_constraint;}
        }
        }
    }
else
    BPs are not reconfigurable;
7   Process end;
```

5 Illustration of the proposed method using the case studies

Five detailed case studies are generated to study the impact of data, gateway and ordering constraints on the reconfigurability property of BPs. All case studies are distinct by the number of BP elements, degree of reconfigurability and complexity level. One case study is discussed in this section and others (library management system, ATM withdrawal systems, hospital management system and online shopping system) are appended in Supplementary materials. One case study is discussed in this section and the other four case studies are explained in supplementary files.

5.1 Online admission system

A college with only the undergraduate (UG) course has got permission to start the postgraduate (PG) course from its new session. Here, previous goal G was to provide education for only UG students. The new goal G' is to provide education to both UG and PG students. Here, a new sub-goal (open course for PG) has been added to the new modified goal G'. Former BP has also been changed into (BP'). The management group can incorporate the PG course with the existing UG course in several different ways.

5.1.1 Data constraint

If UG and PG admission follows the same procedure that is all the activities of the previous and new BPs remain the same, only conditions are different for UG and PG admission, then data constraint is applied. In $BP1$, the activity set contains the following activities, like:

a student registration ($Ac1$)

b check age ($Ac2$)

c show message eligible for UG ($Ac3$)

d check marks ($Ac4$)

e get enrolled ($Ac5$)

f show message not eligible ($Ac6$)

g not enrolled ($Ac7$).

These seven activities remain the same for UG and PG courses. Only, the conditions are changed accordingly. Gateways and events are also the same for both cases. Activities are not changed in $BP2$. Only, a small change has been done in the value range of data. Rule F7 is satisfied with the data constraints. Figures 1(a), 1(b) and 1(c) represent equivalent BP diagrams of admission procedures in undergraduate, postgraduate and combined courses (UG + PG), respectively. In this case, the degree of reconfigurability is $D_{RC} = 0.400$.

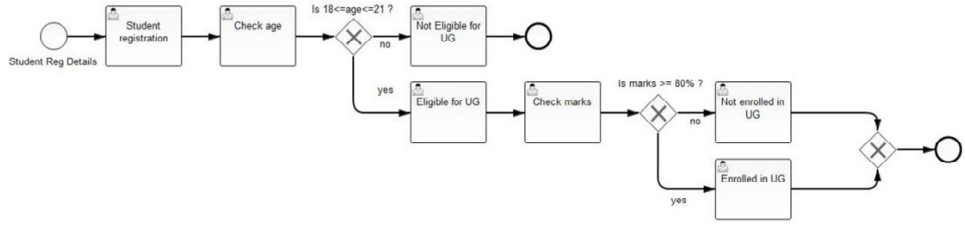
In this case, seven activities, three gateways and two data elements from the pre-existing BP ($BP1$) have been reused in the resultant BP (BP'). Hence, according to the above representation, $D_{RU} = \frac{(7+3+2)}{(11+5+5)} = 12/21 = 0.571 \geq k = 0.3$. So, $BP1$ (online admission system for UG) is reusable in BP'. The list of BP elements that can be reused is [$Ac1$, $Ac2$, $Ac3$, $Ac4$, $Ac5$, $Ac6$, $Ac7$, $Ga1$, $Ga2$, $Ga3$, $Da1$, $Da2$] and $Ga1$ and $Da1$ are modified at the simple (Lw) level.

So, $D_{RA} = \begin{bmatrix} 1 \\ 0 \\ 7 \end{bmatrix}$.

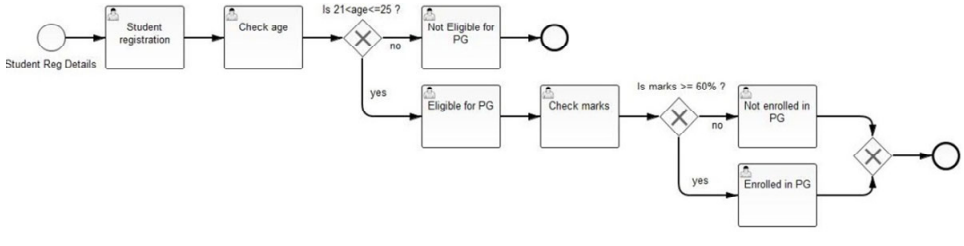
Two Da, four Ac and two Ga with the Hi level modifications, from the newly generated BP ($BP2$) (online admission system for PG) can be reused in BP'.

So, $WM = \begin{bmatrix} 2*3 \\ 4*6 \\ 2*9 \end{bmatrix} = \begin{bmatrix} 6 \\ 24 \\ 18 \end{bmatrix}$. Now the total amount of modification in BP' is $(D_{RA} + WM) = TW = \begin{bmatrix} 1 \\ 0 \\ 7 \end{bmatrix} + \begin{bmatrix} 6 \\ 24 \\ 18 \end{bmatrix} = \begin{bmatrix} 7 \\ 24 \\ 25 \end{bmatrix}$. So, the magnitude of the resultant vector TW is $\sqrt{7^2 + 24^2 + 25^2} = \sqrt{1250} = 35.35$.

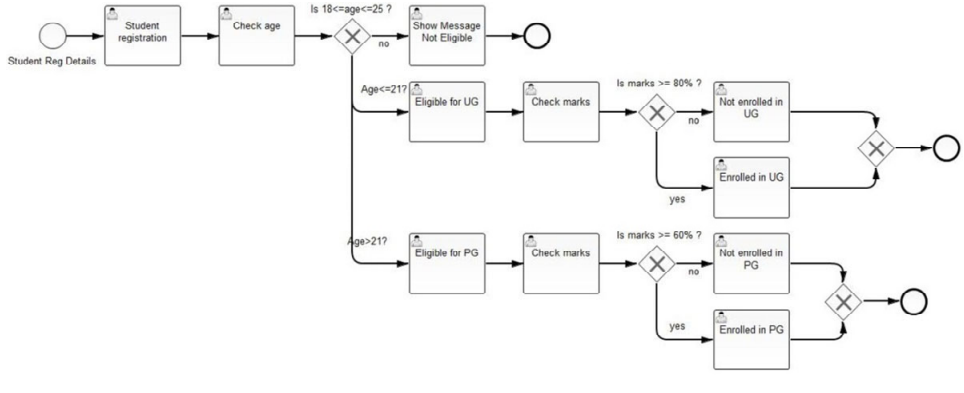
Figure 1 Academic admission procedure BP using data constraint, (a) existing BP for handling only undergraduate admission procedure (input 1) (b) existing BP for handling only postgraduate admission procedure (input 2) (c) reconfigured BP capable of handling both undergraduate and postgraduate admission procedure (output)



(a)



(b)



(c)

If *BP′* would be newly created in its wholeness then the weightage vector *NTW* would be $\begin{bmatrix} 3*3 \\ 11*6 \\ 5*9 \end{bmatrix} = \begin{bmatrix} 9 \\ 66 \\ 45 \end{bmatrix}$. The magnitude of the vector would be $\sqrt{9^2 + 66^2 + 45^2} = \sqrt{6462}$ $= 88.38.$

$$D_{RC} = \frac{35.355}{88.386} = 0.400 \leq x = 0.75$$

Thus, the condition of reconfigurability is satisfied.

5.1.2 Gateway level constraint

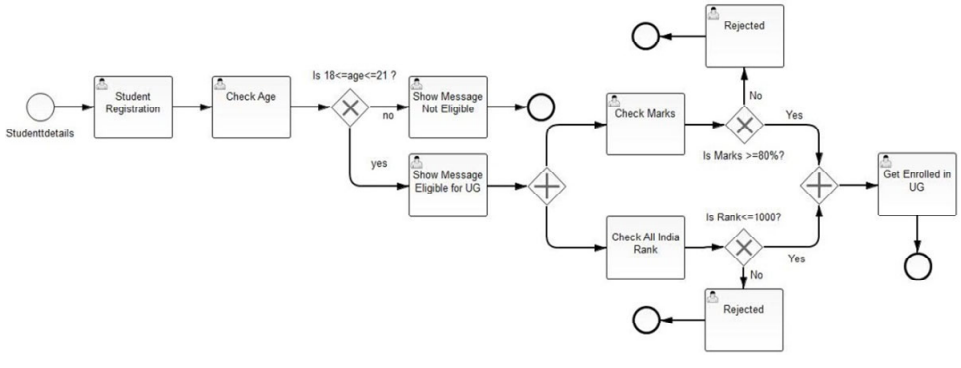
'Check marks' and 'check all India rank' are two mandatory activities for the UG course. Management decides that those two activities are optional for the PG course. In the UG course, parallel gateways are used to express the simultaneous activities, whereas, in the PG course, inclusive gateways are used to represent the optional activities. Data constraints remain the same. Thus, Figures 2(a), 2(b) and 2(c) represent the admission procedure for UG, PG and combined courses, respectively. In this case, the number of elements in previous and new BPs is the same, only the gateway types are changed. Here, the rules F10 and F11 have been satisfied. Here, the degree of reconfigurability value is $D_{RC} = 0.465$.

In this case, nine activities, five gateways and three data elements from the pre-existing BP (*BP*1) have been reused in the resultant BP (*BP′*).

$$D_{RU} = \frac{(9+5+3)}{(15+9+7)} = {}^{17}\!/_{31} = 0.548 \geq k = 0.3.$$

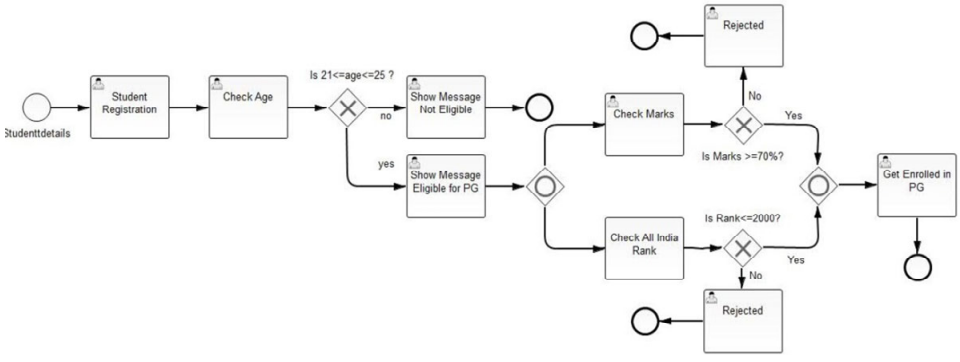
So, *BP*1 (online admission system for UG) is reusable in *BP′*. The list of BP elements that can be reused is [*Ac*1, *Ac*2, *Ac*3, *Ac*4, *Ac*5, *Ac*6, *Ac*7, *Ac*8, *Ac*9, *Ga*1, *Ga*2, *Ga*3, *Ga*4, *Ga*5, *Da*1, *Da*2, *Da*3] and *Ga*1 and *Da*1 are modified at the simple (*Lw*) level.

Figure 2 Academic admission procedure BP using gateway constraint, (a) existing BP for handling only undergraduate admission procedure (input 1) (b) existing BP for handling only postgraduate admission procedure (input 2) (c) reconfigured BP capable of handling both undergraduate and postgraduate admission procedure (output)

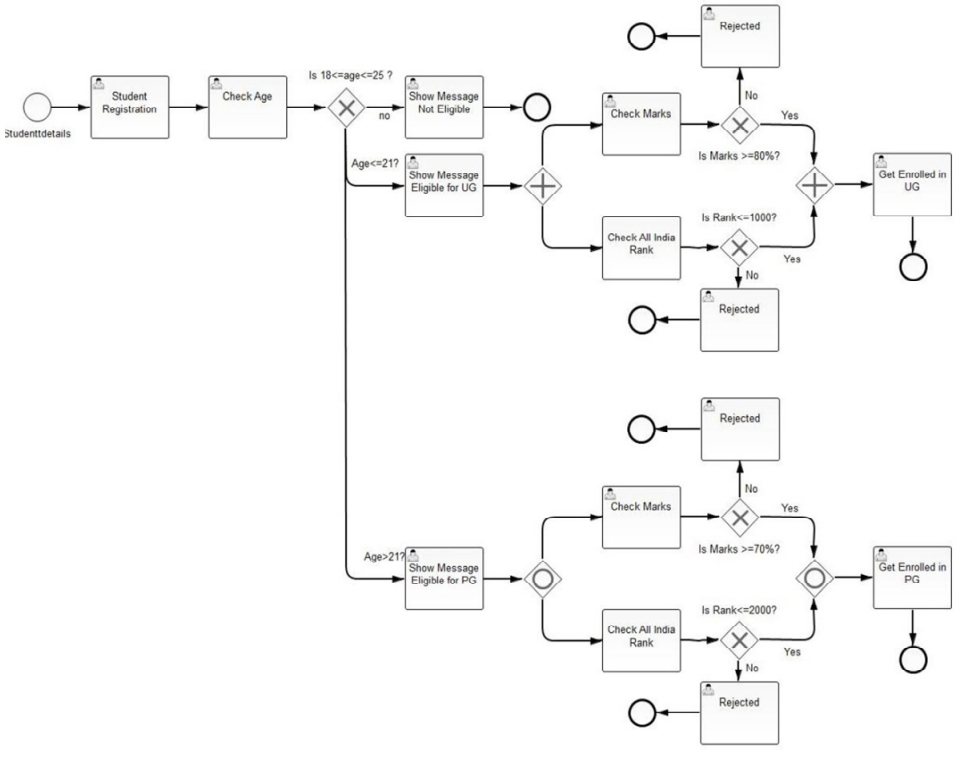


(a)

Figure 2 Academic admission procedure BP using gateway constraint, (a) existing BP for handling only undergraduate admission procedure (input 1) (b) existing BP for handling only postgraduate admission procedure (input 2) (c) reconfigured BP capable of handling both undergraduate and postgraduate admission procedure (output) (continued)



(b)



(c)

So, $D_{RA} = \begin{bmatrix} 1 \\ 0 \\ 7 \end{bmatrix}$.

Three *Da*, six *Ac* and four *Ga* with the *Hi* level modifications, from the newly generated BP (*BP*2) (online admission system for PG) can be reused in *BP′*. So,

$$WM = \begin{bmatrix} 3*3 \\ 6*6 \\ 4*9 \end{bmatrix} = \begin{bmatrix} 9 \\ 36 \\ 36 \end{bmatrix}.$$

Now, the total amount of modification in *BP′* is

$$(D_{RA} + WM) = TW = \begin{bmatrix} 1 \\ 0 \\ 7 \end{bmatrix} + \begin{bmatrix} 9 \\ 36 \\ 36 \end{bmatrix} = \begin{bmatrix} 10 \\ 36 \\ 43 \end{bmatrix}.$$

So, the magnitude of the resultant vector *TW* is $\sqrt{10^2 + 36^2 + 43^2} = \sqrt{3245} = 56.964$.

If *BP′* would be newly created in its wholeness then the weightage vector *NTW* would be $\begin{bmatrix} 6*3 \\ 15*6 \\ 9*9 \end{bmatrix} = \begin{bmatrix} 18 \\ 90 \\ 81 \end{bmatrix}$.

The magnitude of the vector would be $\sqrt{18^2 + 90^2 + 81^2} = \sqrt{14985} = 122.413$.

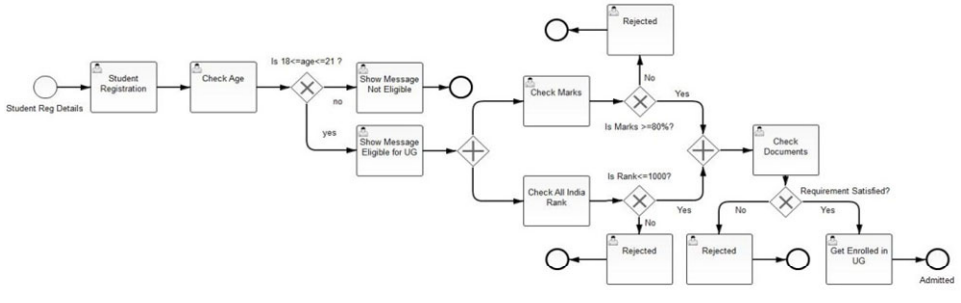
$$D_{RC} = \frac{56.964}{122.413} = 0.465 \leq x = 0.75.$$

Thus, the condition of reconfigurability is satisfied.

5.1.3 Ordering level constraint

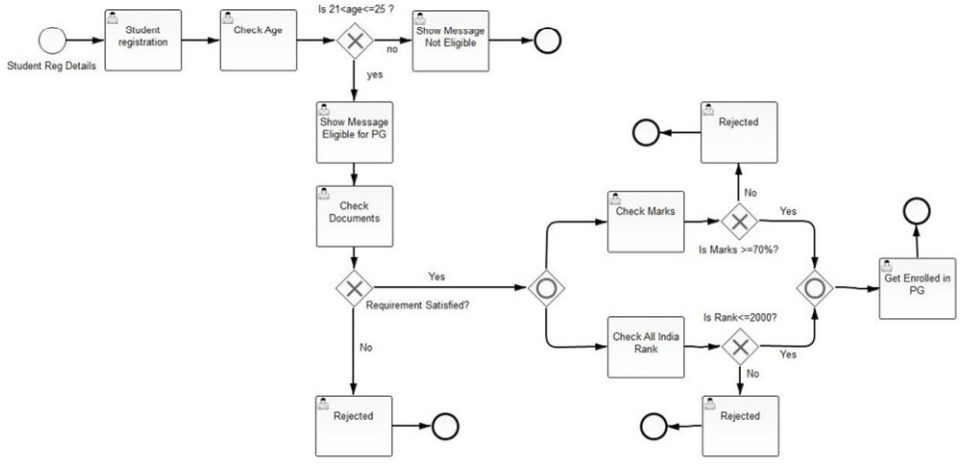
For the UG course, document verification activity is happened, after completion of checking marks and checking all India rank activities. But in the case of PG admission, document verification is happened before checking marks and checking all India rank. So, by only changing the ordering of activities, the modified sub-goal can be achieved. Here, rule F14 is satisfied.

Figure 3 Academic admission procedure BP using ordering constraint, (a) existing BP for handling only UG admission procedure (input 1) (b) existing BP for handling only PG admission procedure (input 2) (c) reconfigured BP for handling both UG and PG admission procedure (output)

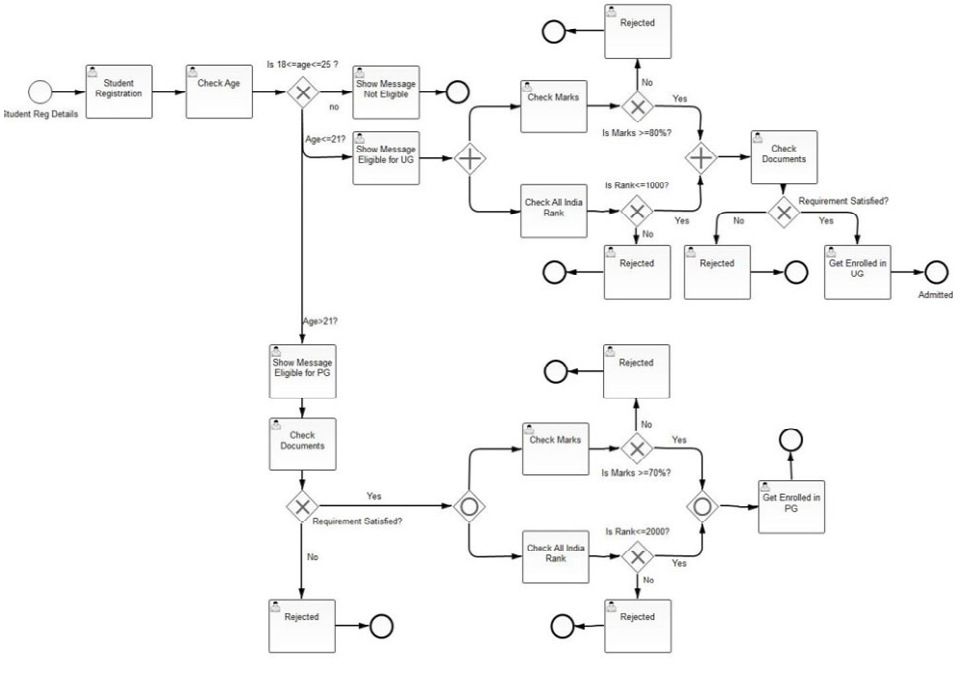


(a)

Figure 3 Academic admission procedure BP using ordering constraint, (a) existing BP for handling only UG admission procedure (input 1) (b) existing BP for handling only PG admission procedure (input 2) (c) reconfigured BP for handling both UG and PG admission procedure (output) (continued)



(b)



(c)

Figures 3(a), 3(b) and 3(c) represent equivalent BP diagram of UG admission procedure, PG admission procedure and combined (UG + PG) admission procedure, respectively. Here, the degree of reconfigurability value is $D_{RC} = 0.471$.

In this case study, whenever data constraint, gateway constraint and ordering constraint need to be changed simultaneously, then data constraint is given to the highest priority as discussed above. At first, the student information (dataset) is categorised into two classes, one is for UG students, and the other class is for the PG students. After the classifications of data, different gateways are chosen to make the decision point, from where the BP flow diverges. Then, the order of execution of different activities is decided. Finally, the decision of which role will execute which set of activities will be taken.

In this case, 11 activities, six gateways and four data elements from the pre-existing BP ($BP1$) have been reused in the resultant BP (BP'). $D_{RU} = \frac{(11+6+4)}{(19+11+9)} = 21/39 = 0.538 \geq k = 0.3$. So, $BP1$ (online admission system for UG) is reusable in BP'. The list of BP elements that can be reused is [$Ac1$, $Ac2$, $Ac3$, $Ac4$, $Ac5$, $Ac6$, $Ac7$, $Ac8$, $Ac9$, $Ac10$, $Ac11$, $Ga1$, $Ga2$, $Ga3$, $Ga4$, $Ga5$, $Ga6$, $Da1$, $Da2$, $Da3$, $Da4$] and $Ga1$ and $Da1$ are modified at the simple (Lw) level. So, $D_{RA} = \begin{bmatrix} 1 \\ 0 \\ 7 \end{bmatrix}$.

Four Da, eight Ac and five Ga with the Hi level modifications, from the newly generated BP ($BP2$) (online admission system for PG) can be reused in BP'. So, $WM = \begin{bmatrix} 4*3 \\ 8*6 \\ 5*9 \end{bmatrix} = \begin{bmatrix} 12 \\ 48 \\ 45 \end{bmatrix}$. Now, the total amount of modification in BP' is

$$(D_{RA} + WM) = TW = \begin{bmatrix} 1 \\ 0 \\ 7 \end{bmatrix} + \begin{bmatrix} 12 \\ 48 \\ 45 \end{bmatrix} = \begin{bmatrix} 13 \\ 48 \\ 52 \end{bmatrix}.$$

So, the magnitude of the resultant vector TW is $\sqrt{13^2 + 48^2 + 52^2} = \sqrt{5177} = 71.95$.

If BP' would be newly created in its wholeness then the weightage vector NTW would be $\begin{bmatrix} 7*3 \\ 19*6 \\ 11*9 \end{bmatrix} = \begin{bmatrix} 21 \\ 114 \\ 99 \end{bmatrix}$.

The magnitude of the vector would be $\sqrt{21^2 + 114^2 + 99^2} = \sqrt{23238} = 152.44$.

$$D_{RC} = \frac{71.951}{152.440} = 0.471 \leq x = 0.75.$$

Thus, the condition of reconfigurability is satisfied.

6 Experimental result and analysis

Detailed performance analysis of the proposed algorithm has been done in this section. The variation of the D_{RC} values with the BPs of different complexity in terms of the

number of BP elements they have and the total execution time (E_t) taken by this proposed algorithm to reconfigure those BPs have been discussed here. Certain benchmark has been set for this experiment, like D_{RU} and D_{RC} values should be less than 1. The proposed algorithm (*RCBPA*) has been implemented through five different case studies containing a different number of gateways, data and user tasks (activities). For each case study, the D_{RC} and the corresponding execution time (E_t) for data, order and gateway constraints have been calculated individually. The implementation of the algorithm has been done on the Java Eclipse platform using document object model (DOM) (Yang et al., 2015; XML DOM, https://www.w3schools.com/xml/dom_intro.asp/), Parser (used to parse XML document) (Chinos and Trombetta, 2009; Kurz, 2016; XML DOM, https://www.w3schools.com/xml/dom_intro.asp/). Camunda BPMN Modeler Camunda Docs, https://docs.camunda.org/get-started/quick-start/; BPMN Tutorial, https://camunda.com/bpmn/) has been used for creating input files with BPMN diagrams. A lower D_{RC} value signifies that the amount of changes required to be done for the conversion of the pre-existing BP into the reconfigurable BP is minimum. On the contrary, the higher value of D_{RC} implies larger overall changes in the pre-existing BP. Consequently, a lesser D_{RC} value shows that the complexity level in terms of the number of BP elements, to be incorporated in the earlier BP is low while this number of BP elements increases with the increment of D_{RC} value. The experimental result also shows that D_{RC} is reduced with the addition of more BP elements to the pre-existing BP. The execution time is measured by setting a counter variable to store the difference between the starting and the ending time of the program execution.

6.1 Comparative study on degree of reconfigurability (D_R) values

To measure the performance of the proposed method, five selected case studies with different levels of complexities have been considered. Table 1 exhibits the degree of reconfigurability (D_R) values in the case of five different real-life scenarios. D_{RC} of each of these five case studies has been calculated for data constraints, gateway constraints and order constraints. From the experimental result, it is observed that more complex BP leads to a high D_{RC} value.

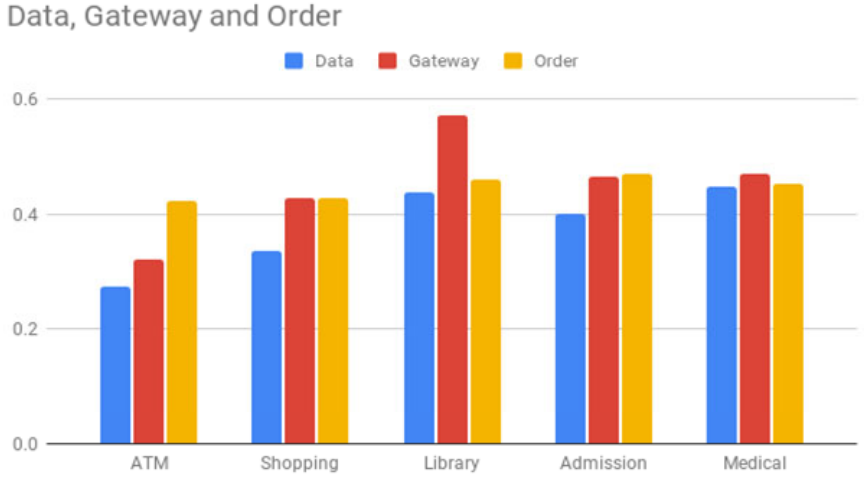
Table 1 Different degree of reconfigurability (D_{RC}) values for corresponding data, gateway and order constraints for five distinct case studies

	ATM	*Online shopping*	*Library*	*Admission*	*Hospital*
Data constraint	0.274	0.336	0.437	0.400	0.447
Gateway constraint	0.322	0.429	0.573	0.465	0.470
Ordering constraint	0.423	0.429	0.460	0.471	0.452

The complexity of a BP is measured by the number of BP elements (user tasks, gateways and datasets) it contains. In the case of data-constraint-based reconfigurable BP, the number of new BP elements to be added is less as compared to the gateway and order-based reconfigurable BPs. It results in a low D_{RC} value for the data constraint-based reconfigurable mechanism. Lesser value of D_{RC} signifies that the level of complexity in terms of the number of new BP elements, needed to be incorporated within the new modified BP is low. D_{RC} value increases with the increment of the number of BP elements that are required to be integrated into the new BP. Here, the lowermost value of

D_{RC} is obtained from the case studies is 0.33. Figure 4 is the graphical representation of distinct D_{RC} values for data, gateway and order constraints for five case studies.

Figure 4 Graphical representation of D_{RC} values corresponding to five case studies (see online version for colours)



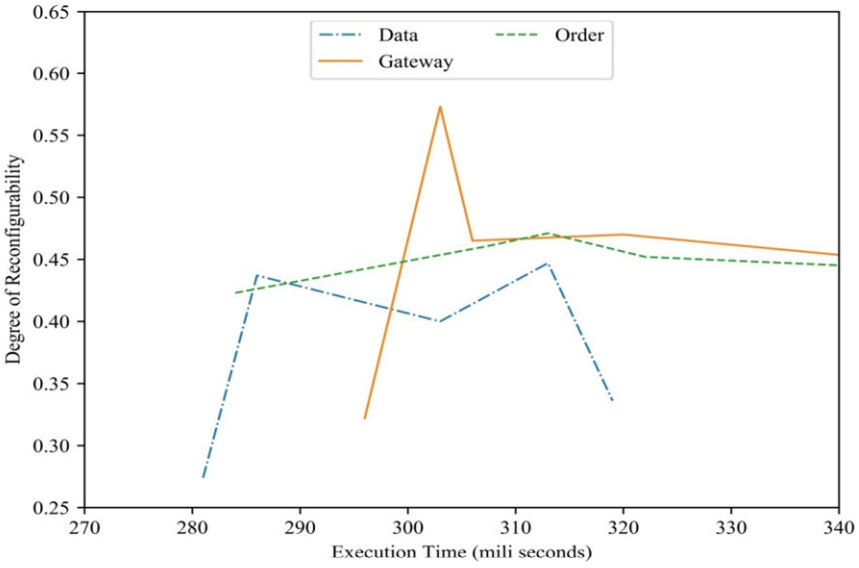
6.2 Comparative study on execution time (E_t)

Table 2 represents execution times taken by the proposed algorithm that implements data, gateway and order constraint-based reconfigurable mechanism in different real-life scenarios. In each of the case study, it is shown that the execution time (E_t) of the proposed algorithm is of the lowest value in the data constraint, while for the gateway and the ordering constraints, the E_t values differ (in several cases, E_t value of gateway constraint is higher than E_t value of ordering constraint and in the other cases the reverse result occurs) depending on the amount of changes required to be done in the pre-existing BP. Figure 5 exhibits that there is no direct relationship between the D_{RC} value and the E_t value because the D_{RC} value signifies the amount of modifications required in the pre-existing BP, while E_t represents the total processing increases with the increasing value of the execution time of that algorithm. It signifies that the incorporation of more BP elements to the existing BP needs more execution time because this integration raises the complexity level of the BP.

Table 2 Execution times (E_t) (ms) for data, gateway and order constraints for five case studies

	ATM	*Online shopping*	*Library*	*Admission*	*Hospital*
Data constraint	281	283	286	303	313
Gateway constraint	296	300	303	306	320
Ordering constraint	284	303	307	313	322

Figure 5 Graphical representation of execution time E_t (ms) versus D_{RC} for data, gateway and order constraints for five case studies (see online version for colours)



6.3 Analysis of results

Table 1 and Table 2 display how the D_{RC} values and corresponding execution times (E_t) differ for five distinct case studies. Those five real-life scenarios deal with BPs having a different number of BP elements and various levels of complexity. In Table 1, D_{RC} value varies from 0.274 to 0.573. This range satisfies the condition that was discussed in Section 3.2. Thus, for each case study, the D_{RC} value is within the specified range, which implies that each BP is reconfigurable. Keeping the overall goal unchanged only a sub-goal is added in every situation. That additional sub-goal causes minimal changes to the existing BP elements. The incorporation of fewer BP elements results in a lower value of D_{RC}. Furthermore, D_{RC} values for data constraints obtain the lowest value amongst the three constraints for individual cases. For the case of gateway constraint and order constraint, the D_{RC} value increases as the result of the insertion of more BP elements. A gateway diverge the process flow among more than one path. Replacement of pre-existing gateway with new one results in the inclusion of several new BP elements. On the other hand, the ordering constraint introduces the changes in the execution order of the user tasks within an existing BP. These modifications in execution order also cause to include multiple new BP elements. Thus, the addition of more BP elements increases D_{RC} value in both constraints (gateway and order) in comparison with data constraints.

The experimental result shows that the value of D_{RC} for the gateway constraint is higher than the D_{RC} value of ordering constraint for the case studies, like library management system and hospital management system while the inverse result is obtained in the other two case studies like ATM withdrawal system and online admission system. For online shopping system, the D_{RC} value possesses the same value. So, in reality, values of D_{RC} for gateway constraint and ordering constraint differ to some extent depending on the complexity level of the BP.

Table 2 shows that the execution time (E_t) value varies from 264 ms to 322 ms. The E_t value includes the processing time for taking two input files containing *BP*1 and *BP*2, the reconfiguration time and the generation time of output as the reconfigured BP (*BP*′). Consequently, E_t value depends on the size (in terms of the number of datasets, activities and gateways) of both BPs (*BP*1 and *BP*2), the number of *Da*, *Ac* and *Ga* elements, taken from *BP*2 and integrated into the pre-existing BP (*BP*1) and the number of total BP elements (including *Da*, *Ac*, *Ga*) in the output BP (*BP*′). So, more complex BP causes an increase in the execution time (E_t). For example, online admission system contains seven *Ac*, three *Ga* and two *Da* in both the input BPs (*BP*1 and *BP*2) and 11 *Ac*, five *Ga* and five *Da* in *BP*′, whereas, ATM withdrawal system contains six *Ac*, two *Ga* and two *Da* in both the input BPs (*BP*1 and *BP*2) and eight *Ac*, two *Ga* and four *Da* in *BP*′. Henceforth, the E_t values for data constraint in the online admission system and the library management system are 312 ms and 281 ms respectively because of the greater and lesser number of BP elements.

The assumption has been made that these five case studies with different complexity levels have different numbers of datasets, user tasks and gateways. The case study on online admission system deals with seven user tasks, three gateways and two datasets while the ATM withdrawal system works with six user tasks, two gateways and two datasets. Consequently, the D_{RC} value for data constraint in the previous case is greater (D_{RC} = 0.400) than the D_{RC} value (D_{RC} = 0.274) of the ATM withdrawal system. Similarly, D_{RC} values for gateway and order constraints in the case of online admission procedure are also greater than the corresponding values of D_{RC} for ATM withdrawal system.

DOM Parser has been chosen for its several advantageous properties like platform and language independence, high navigation abilities and dynamicity. According to the proposed algorithm, two input files (*BP*1 and *BP*2) are represented as the BPMN files and their back end are represented as the XML files. DOM Parser takes those input files and generates the parse trees so that the navigation, the access to any node and the modification to that node of those parse trees can be done very quickly and easily. Moreover, DOM application programming interface (API) is very easy to use because it supports both read and write operations in the XML file and the capability of performing both operations like, read (for processing the input files) and write (for the generation of the output file) operation is the basic requirement of that algorithm. Similarly, Camunda BPMN Modeler provides a REST API that allows users to use any programming language.

7 Conclusions

Reconfigurable BP is the major focus of this paper. The reconfigurability property of a BP makes it adaptable to changes in customer requirements and business strategies of an organisation. When a goal or a sub-goal is modified due to such frequent changes in business requirements, there is no need to replace the previous BP with a new BP. Instead, the earlier version of BP will be reconfigured by rearrangement, addition, or deletion of the BP elements of that BP.

Thus, the elements of BP either can be reused with their original form or can be used in the new BP with minimal changes in their earlier form. A BP is reconfigurable within the user-defined limit, and beyond that limit, the BP will be not reconfigurable, rather it

will be replaceable with a new BP. Here, a systematic approach has been proposed to achieve reconfigurability, based on different types of constraints.

The constraint-based reconfigurability property of a BP allows the users to define the boundary values of the domain of that BP. Henceforth, users can change these limiting values depending on the current business scenarios. Moreover, in this approach, most of the existing BP elements can be used again in the new BP. This reusability property of BP elements reduces the development cost and maintenance cost of the BP. Another benefit of this constraint-based approach is that the BPs can easily adapt to the rapid and unpredictable changes in the internal and external business environment. So, flexible BPs can easily satisfy business requirements. Thus, the constraint-based reconfigurable approach makes BPs more user-friendly, reusable, flexible and cost-effective. Constraints like data constraints, gateway constraints and ordering constraints are demonstrated formally. In addition, equivalent BP diagram representations have also been generated through five case studies. Finally, an algorithm has been proposed and implemented to describe the mechanism of achieving reconfigurability in BPs.

All these BPs that are used to represent the case studies satisfy the specified range of D_{RC}. Consequently, all BPs are reconfigurable and the D_{RC} value for data constraint obtains the highest value among three constraints for each of the five case studies. It is also observed from the experimental result, that the D_{RC} value decreases with the increase of the complexity level of a BP. In addition, more complex BP needs more execution time (E_t) to reconfigure.

The future work will be focused on the enrichment of three software metrics (D_{RU}, D_{RA} and D_{RC}) by the inclusion of more properties of the BP elements within those metrics. Those properties may include the type of BP elements like different weightage values may be given to the different types of gateways like exclusive gateways, inclusive gateways and parallel gateways based on their modification effect on the flow of a BP. Similarly, the number of branches, associated with the gateways will also be assigned the different weightage values such as higher weightage value should be given to the gateways having more number of branches due to their greater impact on the BP flow. Another aspect to be considered is the inclusion of the role constraint within a BP. There should be a proper allocation of the set of activities to its corresponding role to determine the execution path of a BP.

Appendices/Supplementary materials are available on request by emailing the corresponding author or can be obtained under https://tinyurl.com/5xrpu9ua.

References

Ardagna, D. and Pernici, B. (2007) ‘Adaptive service composition in flexible processes’, *IEEE Transactions on Software Engineering*, Vol. 33, No. 6, pp.369–384.

Arsanjani, A. and Ng, D. (2002) ‘Business compilers: towards supporting a highly re-configurable architectural style for service-oriented architecture’, in *Companion of the 17th Annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications*, pp.26–27.

Basias, N., Themistocleous, M. and Morabito, V. (2013) ‘SOA adoption in e-banking’, *Journal of Enterprise Information Management*, Vol. 26, No. 6, pp.719–739.

Bazoun, H., Bouanan, Y., Zacharewicz, G., Ducq, Y. and Boye, H. (2014) 'Business process simulation: transformation of BPMN 2.0 to DEVS models (WIP)', in *Proceedings of the Symposium on Theory of Modeling & Simulation – DEVS Integrative*, pp.1–7.

BPMN Tutorial, *BPMN 2.0 Tutorial for Beginners – Learn BPMN Camunda BPM* [online] https://camunda.com/bpmn/ (accessed 24 January 2020).

Camunda Docs, *Get Started with Camunda* [online] https://docs.camunda.org/get-started/quick-start/ (accessed 24 January 2020).

Chakraborty, P. and Sarkar, A. (2017) 'Context driven approach for enterprise architecture framework', in *Proceedings of 16th International Conference on Computer Information Systems and Industrial Management Applications*, Springer, Bialystok, Poland, pp.277–289.

Chakraborty, P. and Sarkar, A. (2019) 'Score framework: a layered approach for enterprise architecture', *International Journal of Business and Systems Research*, Vol. 13, No. 4, pp.438–467.

Chinos, M. and Trombetta, A. (2009) 'Modeling and validating BPMN diagrams', in *IEEE Conference on Commerce and Enterprise Computing*, IEEE, Vienna, Austria, pp.353–360.

Gao, H. and Miao, H. (2013) 'A quantitative model-based selection of web service reconfiguration', in *2013 14th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, IEEE, pp.365–371.

Gao, X., Li, Z., Zhao, L. and Yao, Y. (2006) 'Reconfiguring business process for enterprise information system based on UML and polychromatic sets', in *Research and Practical Issues of Enterprise Information Systems*, Springer, Vienna, Austria, pp.371–381.

Hermosillo, G., Seinturier, L. and Duchien, L. (2010) 'Using complex event processing for dynamic business process adaptation', in *2010 IEEE International Conference on Services Computing*, IEEE, pp.466–473.

Jabbar, ZA., Kumar, M. and Samreen, A. (2015) 'Designing conceptual framework for aligning service oriented architecture with business process', *Journal of Computer and Communications*, Vol. 3, No. 11, pp.11–22.

Kurz, M. (2016) 'Automating BPMN interchange testing' in *42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, IEEE, Limassol, Cyprus, pp.331–338.

Li, N., Kang, J. and Lv, W. (2005) 'A hybrid approach for dynamic business process mining based on reconfigurable nets and event types', in *IEEE International Conference on e-Business Engineering (ICEBE'05)*, IEEE, pp.289–294.

Li, Y., Zhang, X., Yin, Y. and Lu, Y. (2011) 'Towards functional dynamic reconfiguration for service-based applications', in *2011 IEEE World Congress on Services*, IEEE, Washington, DC, pp.467–473.

Marques, M., Agostinho, C., Zacharewicz, G. and Jardim-Goncalves, R. (2017) 'Reconfigurable and updatable product-service systems: the path for sustainability and personalization', in *Proceedings of the Symposium on Model driven Approaches for Simulation Engineering*, pp.1–12.

Mendonça, D.F., Rodrigues, G.N., Favacho, A. and Holanda, M. (2013) 'A systematic mapping study on service oriented computing in the context of quality of services', in *2013 VII Brazilian Symposium on Software Components, Architectures and Reuse*, IEEE, Brasilia, Brazil, pp.39–48.

Monk, E. and Wagner, B. (2012) *Concepts in Enterprise Resource Planning*, Cengage Learning, Boston, MA, USA.

Sarno, R., Pamungkas, E.W. and Sunaryono, D. (2015) 'Business process composition based on meta models', in *2015 International Seminar on Intelligent Technology and its Applications (ISITIA)*, IEEE, pp.315–318.

Silva, R.M.D., Junqueira, F., Filho, D.J.S. and Miyagi, P.E. (2016) 'Control architecture and design method of reconfigurable manufacturing systems', *Control Engineering Practice*, Vol. 49, No. 1, pp.87–100.

Xiao, Z., Cao, D., You, C. and Mei, H. (2011) 'Towards a constraint-based framework for dynamic business process adaptation', in *2011 IEEE International Conference on Services Computing*, IEEE, pp.685–692.

XML DOM, *XML DOM Tutorial* [online] https://www.w3schools.com/xml/dom_intro.asp/ (accessed 12 February 2020).

Yang, D., Wei, Z. and Yang, Y. (2015) 'A novel implementation of a hash function based on XML DOM Parser', in *International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, IEEE, Xi'an, China, pp.5–8.

Yin, J., Luo, Z., Li, Y. and Wu, Z. (2017) 'Service pattern: an integrated business process model for modern service industry', *IEEE Transactions on Services Computing*, Vol. 10, No. 6, pp.841–853.

Yu, T. and Lin, K. (2005) 'Adaptive algorithms for finding replacement services in autonomic distributed business processes', in *Proceedings Autonomous Decentralized Systems*, IEEE, pp.427–434.

Zhang, J., Lee, J. and Lin, K. (2012) 'Context-aware proactive process reconfiguration in service-oriented architecture', in *2012 IEEE 14th International Conference on Commerce and Enterprise Computing*, IEEE, pp.62–69.

Zhang, Q., Zou, Y., Tong, T., McKegney, R. and Hawkins, J. (2005) 'Automated workplace design and reconfiguration for evolving business processes', in *CASCON*, Citeseer, pp.320–333.