Relationship-Preserving Change Propagation in Process Ecosystems

Tri A. Kurniawan*, Aditya K. Ghose, Hoa Khanh Dam, and Lam-Son Lê

Decision Systems Lab., School of Computer Science and Software Engineering,
University of Wollongong, NSW 2522, Australia
{tak976,aditya,hoa,lle}@uow.edu.au

Abstract. As process-orientation continues to be broadly adopted – evidenced by the increasing number of large business process repositories, managing changes in such complex repositories becomes a growing issue. A critical aspect in evolving business processes is change propagation: given a set of primary changes made to a process in a repository, what additional changes are needed to maintain consistency of relationships between various processes in the repository. In this paper, we view a collection of interrelated processes as an ecosystem in which inter-process relationships are formally defined through their annotated semantic effects. We also argue that change propagation is in fact the process of restoring consistency-equilibrium of a process ecosystem. In addition, the underlying change propagation mechanism of our framework is leveraged upon the well-known Constraint Satisfaction Problem (CSP) technology. Our initial experimental results indicate the efficiency of our approach in propagating changes within medium-sized process repositories.

Keywords: inter-process relationship, semantic effect, process ecosystem, change propagation, constraint network.

1 Introduction

Nowadays, modeling and managing business processes is an important approach for managing organizations from an operational perspective. In fact, a recent study [11] has shown that the business process management (BPM) software market reached nearly $1.7 billion in total software revenue in 2006 and this number continues to grow. In addition, today's medium to large organizations may have collections of hundreds or even thousands of business process models (e.g. 6,000+ process models in Suncorp's process model repository for insurance [16]). Therefore, it becomes increasingly critical for those organizations to effectively manage such large business process repositories.

In recent years, the ever-changing business environment demands an organization to continue improving and evolving its business processes to remain competitive. As a result, the most challenging aspect of managing a repository of business processes is dealing with changes. Since business processes within

* On leave from a lecturership at University of Brawijaya, East Java, Indonesia

C. Liu et al. (Eds.): ICSOC 2012, LNCS 7636, pp. 63–78, 2012.

a repository can be inter-dependent (in terms of both design and execution), changes to one process can potentially have impact on a range of other processes. For example, changes initially made to a sub-process (e.g. removing an activity) may lead to secondary changes made to the processes that contain this sub-process. Such changes may lead to further changes in other dependent processes. The ripple effect that an initial change may cause in a process repository is termed *change propagation*. In a large business process repository, it becomes costly and labor intensive to correctly propagate changes to maintain the consistency among the inter-dependent process models. Therefore, there is an emerging need for techniques and tools that provide more effective (semi-)automated support for change propagation within a complex process repository.

There has been however very little work on supporting change propagation in process model collections [7]. Our proposed framework aims to fill that gap. We view a collection of interrelated process models as an ecosystem [9]. In such ecosystem, process models play a role analogous to that of biological entities in a biological ecosystem. They are created (or discovered, using automated toolkits [10]), constantly changed during their lifetimes, and eventually discarded. Changes made to a process may cause perturbations (i.e. inconsistencies) in the ecosystem in the form of critical inter-process relationships being violated. In this view, a process ecosystem is considered to be in an (consistency-)equilibrium if its all inter-process relationships are mutually consistent. Change propagation is therefore reduced to finding an equilibrium in a process ecosystem.

We further view the problem of finding an equilibrium in a process ecosystem as a constraint satisfaction problem (CSP) in which each process model is mapped to a node and each relationship (between two process models) is a constraint (between the corresponding nodes) in a CSP. This paper is also built on top of our previous work [15], which provides formal definitions for three common types of inter-process relationships (namely part-whole, generalization-specialization and inter-operation) based on concepts of semantic effect-annotated business processes [12]. Specifically, in this paper we propose a machinery to automatically establish relationships between process models in a process repository based on these formalizations. Based on these established relationships, we construct a constraint network [6] of a process ecosystem containing a violated relationship. Candidate values for each individual process node in a CSP can be obtained from the redesign of the process, which can be implemented using existing business process redesign approaches (see, e.g. [13, 19]) or manually produced by the analysts. The CSP encoding allows us to plug different individual process redesign modules without affecting the remaining parts of the architecture.

The rest of the paper is organized as follows. Sec. 2 briefly describes semantic effect-annotated process model and inter-process relationships. Sec. 3 explains how such relationships can be established. Sec. 4 proposes our approach to preserve such relationships in propagating changes within a process ecosystem. In Sec. 5, we present an empirical validation of our approach. We then discuss related work in Sec. 6, and conclude and layout some future work in Sec. 7.

2 Preliminaries

Semantic Effect-Annotated Process Model. An effect annotation relates to a particular result or outcome to an activity in a process model [14]. An activity represents the work performed within a process. Activities are either atomic (called a *task*, i.e. they are at the lowest level of detail presented in the diagram and can not be further broken down) or compound (called a *sub-process*, i.e. they can be broken down to see another level of process below) [21]. In an annotated BPMN process model, as our approach relies on, we annotate each activity with its (immediate) effects. We define these immediate effects as the immediate results or outcomes of executing an activity in a process model. We consider that multiple effects can be immediately resulted in such execution. We shall leverage the ProcessSEER [12] tool to annotate process model with the semantic effects. This annotation allows us to determine, at design time, the effects of the process if it were to be executed up to a certain point in the model. These effects are necessarily non-deterministic, since a process might have taken one of many possible alternative paths through a process design to get to that point. We define a procedure for *pair-wise effect accumulation*, which, given an ordered pair of activities with effect annotations, determines the cumulative effect after both activities have been executed in contiguous sequence.

Let t_i and t_j be an ordered pair of activities connected by a sequence flow such that t_i precedes t_j. Let $e_i = \{c_{i1}, \ldots, c_{im}\}$ and $e_j = \{c_{j1}, \ldots, c_{jn}\}$ be the corresponding pair of effect annotations, respectively. If $e_i \cup e_j$ is consistent, then the resulting cumulative effect is $e_i \cup e_j$. Otherwise, we define $e'_i = \{c_k\}$ where $c_k \in e_i$ and $\{c_k\} \cup e_j$ *is consistent*, and the resulting cumulative effect to be $e'_i \cup e_j$. In the following, we shall use $ACC\,(e_p, e_q)$ to denote the result of pair-wise effect accumulation of two contiguous activities t_p and t_q with the immediate effects e_p and e_q, respectively.

Effects are only accumulated within participant lanes (i.e. role represented as a pool) and are not including inter-participant within inter-operation business process. In addition to the effect annotation of each activity, we also denote E_t as the cumulative effect of activity t. E_t is defined as a set $\{es_{t1}, \ldots, es_{tm}\}$ of alternative *effect scenarios* based on the $1, \ldots, m$ alternative paths reaching the activity. Alternative effect scenarios are introduced by AND-joins or XOR-joins or OR-joins. We accumulate effects through a left-to-right pass of a participant lane, applying the pair-wise effect accumulation procedure on contiguous pairs of activities connected via control flow links. The process continues without modification over splits. Joins require special consideration. In the following, we describe the procedure to be followed in the case of 2-way joins only, for brevity. The procedure generalizes in a straightforward manner for n-way joins.

In the following, let t_p and t_q be two activities immediately preceding a join. Let their cumulative effect annotations be $E_p = \{es_{p1}, \ldots, es_{pm}\}$ and $E_q = \{es_{q1}, \ldots, es_{qn}\}$, respectively. Let e be immediate effect and E be cumulative effect of an activity t immediately following the join.

For **AND-joins**, we define $E = \{ACC\,(es_{pi}, e) \cup ACC\,(es_{qj}, e)\}$ where $es_{pi} \in E_p$ and $es_{qj} \in E_q$. Note that we do not consider the possibility of a pair of

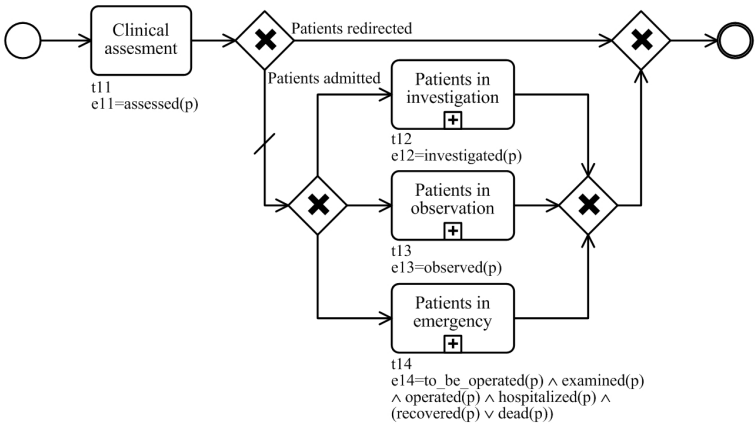


Fig. 1. *Management of patients on arrival* process

effect scenarios es_{pi} and es_{qj} being inconsistent, since this would only happen in the case of intrinsically and obviously erroneously constructed process models. The result of effect accumulation in the setting described here is denoted by $ANDacc\,(E_p, E_q, e)$. For **XOR-joins**, we define $E = \{ACC\,(es_r, e)\}$ where $es_r \in E_p$ or $es_r \in E_2$. The result of effect accumulation in the setting described here is denoted by $XORacc\,(E_p, E_q, e)$. For **OR-joins**, the result of effect accumulation in such setting is denoted by $ORacc\,(E_p, E_q, e) = ANDacc\,(E_p, E_q, e) \cup XORacc\,(E_p, E_q, e)$.

Figure 1 illustrates a semantic effect-annotated BPMN process model. The immediate effect e_i of each activity t_i is represented in a Conjunctive Normal Form (CNF) allowing us to describe such effect as a set of outcome clauses. Let p be patient to be observed and treated. For example, activity t_{13} has an immediate effect $e_{13} = observed(p)$ which depicts the outcomes of executing such activity. The cumulative effects of execution the process until t_{13} can be computed by accumulating the effects starting from t_{11} until t_{13}, i.e. $assessed(p) \wedge observed(p)$. We can also compute for the other activities in a similar way.

Inter-process Relationships. We recap relationships formalization described in our previous work [15]. We classify these relationships into two categories: *functional dependencies* and *consistency links*. A functional dependency exists between a pair of processes when one process needs support from the other for realizing some of its functionalities. In this category, we define two relationship types, i.e. *part-whole* and *inter-operation*. A consistency link exists between a pair of processes when both of them have intersecting parts which represent the same functionality, i.e. the outcomes of these parts are exactly the same. They are functionally independent. We identify one type in this category, i.e. *generalization-specialization*. Our framework focuses on these three types.

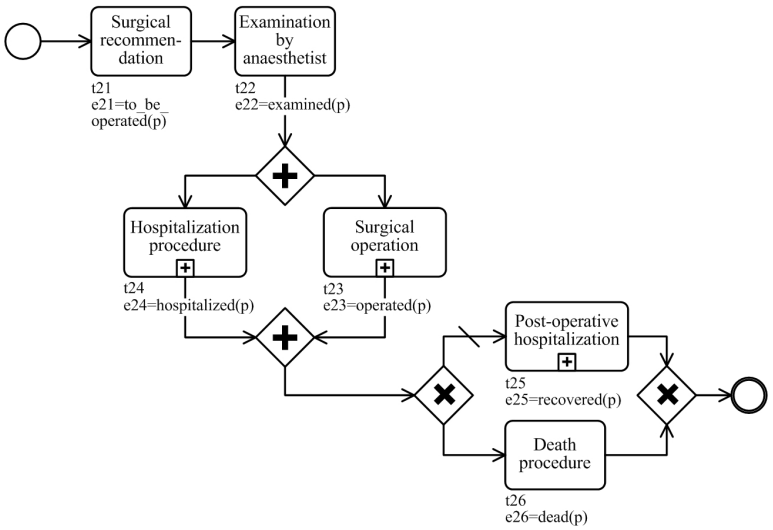


Fig. 2. Expansion of *Patients in emergency* sub-process in Figure 1

We use $acc(P)$ to denote the end cumulative effects of process P; $CE(P, t_i)$ to describe cumulative effect at the point of activity t_i within process P; and es_j to denote an effect scenario *j-th*. Noted, each of $acc(P)$ or $CE(P, t_i)$ is a set of effect scenarios. Each effect scenario is represented as a set of clauses and will be viewed, implicitly, as their conjunction.

(i) Part-Whole. A part-whole relationship exists between two processes when one process is required by the other to fulfill some of its functionalities. More specifically, there must be an activity in the 'whole' process representing the functionalities of the 'part' process. The 'part' process is also commonly referred to as a sub-process within the 'whole' process. Logically, there is an insertion of the functionalities of the 'part' into the 'whole'.

We define the insertion of process $P2$ in process $P1$ at activity t, $P1 \uparrow^t P2$, is a process design obtained by viewing $P2$ as the sub-process expansion of t in $P1$. We then define the part-whole as follows: $P2$ is a direct part of $P1$ iff there exists an activity t in $P1$ s.t. $CE(P1, t) = CE(P1 \uparrow^t P2, t)$. Let us consider an example[1] of the part-whole relationship which is illustrated in Figures 1 and 2 (called $P1$ and $P2$, respectively). Such relationship is reflected by the activity *Patients in emergency* (t_{14}) in $P1$ whose functionality represents $P2$. This means that the result of executing activity t_{14} in $P1$ is solely the result of executing $P2$, and vice-versa. The insertion point here is at t_{14} in $P1$. The cumulative effect of $P1$ at this point is $CE(P1, t_{14}) = \{es_{14}\}$; $es_{14} = assessed(p) \wedge to\_be\_operated(p) \wedge examined(p) \wedge operated(p) \wedge$

[1] We will only exemplify the part-whole relationship due to space limitation. In this paper, we only address direct relationships for the shake of efficiency in propagating the changes. The indirect ones can be referred in our previous work [15].

$hospitalized\,(p) \wedge (recovered\,(p) \vee dead\,(p))$. We only have one effect scenario, i.e. es_{14}. Furthermore, the cumulative effect of $P1$ at t_{14} by inserting $P2$ at this activity is $CE\,(P1 \uparrow^{t_{14}} P2, t_{14}) = assessed\,(p) \wedge to\_be\_operated\,(p) \wedge examined\,(p) \wedge operated\,(p) \wedge hospitalized\,(p) \wedge (recovered\,(p) \vee dead\,(p))$. Finally, we can infer that $P2$ is a part of $P1$ since $CE\,(P1, t_{14}) = CE\,(P1 \uparrow^{t_{14}} P2, t_{14})$.

(ii) Inter-operation. An inter-operation relationship exists between two processes when there is at least one message exchanged between them and there is no cumulative effect contradiction between tasks involved in exchanging messages. Formally, given processes $P1$ and $P2$, an inter-operation relationship exists between them including activities t_i and t_j iff the following holds: (i) $\exists t_i$ in $P1$ $\exists t_j$ in $P2$ such that $t_i \rightharpoonup t_j$ denotes that t_i sends a message to t_j, or $t_j \rightharpoonup t_i$, if the message is in the opposite direction; (ii) let $E_i = \{es_{i1}, es_{i2}, \ldots, es_{im}\}$ be the cumulative effects of process $P1$ at task t_i, i.e. $CE\,(P1, t_i)$, and $E_j = \{es_{j1}, es_{j2}, \ldots, es_{jn}\}$ be the cumulative effects of process $P2$ at task t_j, i.e. $CE\,(P2, t_j)$. Then, there is no contradiction between E_i and E_j for all $es_{ip} \in E_i$ and $es_{jq} \in E_j$ s.t. $es_{ip} \cup es_{jq} \vdash \perp$ does not hold, where $1 \leq p \leq m$ and $1 \leq q \leq n$.

Effect contradiction exists if the expected effects differ from the given effects. If this is the case, we do not consider such relationship as an inter-operation one even though there is a message between both processes.

(iii) Generalization-Specialization. A generalization-specialization relationship exists between two processes when one process becomes the functional extension of the other. More specifically, the *specialized* process has the same functionalities as in the *generalized* one and also extends it with some additional functionalities. One way to extend the functionalities is by adding some additional activities so that the intended end cumulative effects of the process are consequently extended. Another way involves enriching the immediate effects of the existing activities. In this case, the number of activities remains the same for both processes but the capabilities of the *specialized* process is extended. Noted, the *specialized* process inherits all functionalities of the *generalized* process, as formally defined as follows. Given process models $P1$ and $P2$, $P2$ is a specialization of $P1$ iff $\forall es_i \in acc\,(P1)$, $\exists es_j \in acc\,(P2)$ such that $es_j \models es_i$; and $\forall es_j \in acc\,(P2)$, $\exists es_i \in acc\,(P1)$ such that $es_i \models es_j$.

3 Establishing Inter-process Relationships

The formal definitions of inter-process relationships that we discussed in the previous section empower us to systematically specify (manually or automatically) which process models are related to others in a process repository. To do so, analysts who need to manage the large and complex process repositories must effectively explore the space of all possible pairings of process designs to determine what relationships (if any) should hold in each instance. Note that this generates a space of $\binom{n}{2}$ possibilities, where n is the number of distinct process designs in the repository. Clearly, this has the potential to be an error-prone exercise. An analyst may normatively specify a relationship that does not actually

hold (with regard to our definition of the relationship) between a pair of process designs. In some cases, an analyst might need help in deciding what relationship ought to hold. We therefore develop a *user-interactive* machinery to assist analysts in deciding what type of *normative* relationship should hold between a pair of processes. We consider two approaches in such assistance, i.e. *checking* and *generating* modes. In the *checking* mode, we will assess whether a relationship specified by an analyst does indeed hold with regard to our formal definition. If this is the case, the relationship between the two processes can be established. Otherwise, the tool would alert the analyst and also suggest the actual relationship that may be found between the two processes[2]. In the *generating* mode, our machinery systematically goes through all process models in the repository, generates all possible relationships between them, and present these to the analyst for confirmation. Note that the space of alternative relationships can be large, specially in the case of part-whole relationships. For example, given 4 processes $P1, P2, P3$ and $P4$ where $P2$ is part of $P1$, $P3$ is part of $P2$ and $P4$ is part of $P3$. Not only direct relationships, the tool would also suggest all indirect relationships among them, e.g. $P4$ is (indirectly) part of $P1$, $P4$ is (indirectly) part of $P2$. However, these indirect relationships would not be useful to be maintained since change propagation can still be performed through the direct ones. Hence, the decision should be made by the analyst in both approaches.

Once a relationship is established, a *relationship descriptor* is created. Such a descriptor contains details that are relevant to its associated relationship including identities of each pair of processes and their established relationship type. However, a descriptor can also be enriched with any additional information relevant to the existing relationship types, e.g. the insertion point activity in a part-whole relationship. Relationship descriptors are maintained (i.e. created, updated, and removed) during the relationships establishment and maintenance.

The relationship-establishing algorithms for both approaches, require transformation of each process model into a graph, i.e. transforming each activity, gateway and start/end events into a node and each flow into an edge. These algorithms may involve two runs in evaluating a given pair of processes for each relationship type excluding the inter-operation. On the first run, we evaluate the first process with respect to a normative relationship constraint to the second one. If the constraint does hold, we establish the relationship between the two processes. Otherwise, in the second run we evaluate the second process with respect to the constraint to the first one. For example, the part-whole establishment algorithm can be described as follows[3]. The inputs are two process graphs, i.e. denoted pa and pb, and the outputs are either an instance of relationship descriptor or $null$. The algorithm will assess whether pa is part of pb by computing $CE\,(pb, n)$ and $CE\,(pb \uparrow^n pa, n)$ of each sub-process node $n \in pb$. If $CE\,(pb, n) = CE\,(pb \uparrow^n pa, n)$ then it returns an instance of relationship descriptor of such relationship. Otherwise, it returns $null$. On the first run, we evaluate the first process to the second one. If it is not satisfied, we then evaluate the

[2] If it is "unknown" relationship (refers to our formal definitions), it is not maintained.
[3] Algorithms for the two other relationships are not explained due to space limitation.

second process to the first one on the second run. If a given relationship type cannot be established, we continue the evaluation with the other relationships between the processes in consideration. Note that the part-whole relationship evaluation must be performed before the generalization-specialization relationship evaluation, since the former is a special case of the latter. Inappropriate evaluation ordering of these two relationships might have unexpected result, i.e. every part-whole will be suggested as generalization-specialization. There is no evaluation ordering constraint for inter-operation relationship type.

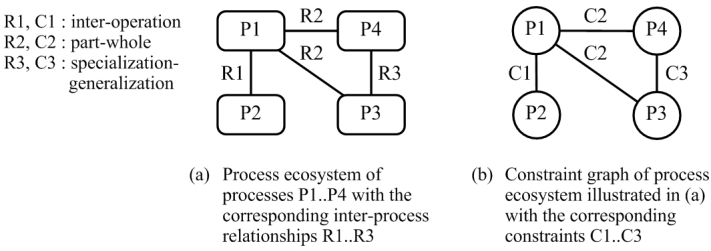
4 Relationship-Preserving Change Propagation

To preserve the established relationships, the changes need to be propagated across different processes. To do so, the process ecosystem containing the initial changes should firstly defined as the boundary of such propagation. We then deal with how to redesign a process for resolving the relationships violations. Further, we apply our CSP approach to find an equilibrium in a process ecosystem.

Process Ecosystem. We view a collection of interrelated process models within a process repository as an ecosystem [9]. However, we further restrict that any process model in a process ecosystem must be traceable to any other process models in the ecosystem. This traceability may involve many relationships with regard to the relationships we formally defined earlier. A process model may have more than one relationship with the others in the ecosystem. In addition, there may be more than one process ecosystem within a process repository. Furthermore, since a change made to a process would only affect other processes in the same ecosystem, we will only propagate changes within this process ecosystem. A process ecosystem is in *(consistency-)equilibrium* if and only if every inter-process relationship in the ecosystem is consistent with our earlier definitions. We shall refer to a process ecosystem that violates the consistency equilibrium condition a *perturbed-equilibrium* ecosystem. Consistency perturbation is often the outcome of change to one or more process models in an ecosystem. Restoring consistency-equilibrium involves making further changes to other process models in the ecosystem and so on, which is, in fact, change propagation. We shall refer to an ecosystem resulted from such restoration a *restored-equilibrium* ecosystem.

Resolving Relationship Violations. To resolve a relationship violation, the processes involving in this relationship need to be changed. There can be many options to do such changes, each of which results in a variant of the original process (called process variant). Generating a process variant for a given process can be done automatically by a machinery[4] or manually by the analyst. The procedure required to resolve a relationship violation between a pair of processes $P1$ and $P2$ due to changes to one of them, e.g. $P1$, can be described as follows.

[4] The techniques for automatically generating all process variants are out of scope of this paper. We leave them as our future investigations. We have used only analyst-mediated process redesign in our implementation and current evaluation.



(a) Process ecosystem of processes P1..P4 with the corresponding inter-process relationships R1..R3

(b) Constraint graph of process ecosystem illustrated in (a) with the corresponding constraints C1..C3

Fig. 3. Transforming a process ecosystem into a constraint graph

We identify such changes in $P1$ that can trigger the violation with regard to our formal definitions. Based on the relationship type, the changes must be propagated to $P2$ to get its variant such that the relationship constraint can be re-satisfied. For example, let $P1$ and $P2$ be the whole and the part, respectively. Let t_i in $P1$, with immediate effects e_{t_i}, be a sub-process representing $P2$ s.t. the condition C is satisfied, i.e. $CE\,(P1, t_i) = CE\,(P1 \uparrow^{t_i} P2, t_i)$. The possible change introduced in $P1$ that can cause violations is a change to t_i, i.e. either by: (i) changing e_{t_i} to be e'_{t_i} s.t. $e_{t_i} \neq e'_{t_i}$ or (ii) dropping t_i. In the first case, we need to change $P2$ to be $P2'$ by either adding or deleting some activities or reducing or extending some immediate effects of some particular activities s.t. C is satisfied with e'_{t_i}. We no longer need to maintain the relationship for the second case. Note that changing $P1$ by excluding t_i will not cause any violation.

CSP in Process Ecosystems. A CSP consists of a set of variables X, for each variable there exists a finite set of possible values D, and a set of constraints C restricting the values that the variables can simultaneously take [2]. Each constraint defines a relation between a subset of variables and constraints the possible values for the involved variables. We consider a constraint involving only one variable as a unary constraint, two variables as a binary, and so on.

A solution to a CSP is an assignment of a value from its domain to every variable, in such a way that every constraint is satisfied. We may want to find only one solution, all solutions or an optimal solution [2]. Solutions to a CSP can be performed by searching systematically for the possible values which can be assigned to a variable. There are generally two search methods. The first method involves either traversing the space of partial solutions [2] (e.g. backtracking, backjumping and backmarking algorithms) or reducing the search space through constraint propagation (i.e. look-ahead). In variable selection, the look-ahead strategy seeks to control the size of the remaining search space. In value selection, it seeks a value that is most likely to lead to a consistent solution. Performing constraint propagation at each variable will result a smaller search space, but the overall cost can be higher since the cost for processing each variable will be more expensive. The second method involves repairing an inconsistent complete assignment/solution (e.g. min-conflicts and repair-based algorithms [17])

Algorithm 1: Generating a restored-equilibrium process ecosystem : *repair* approach

Input:
p, a changed process model
PE_p, graph of a perturbed-equilibrium process ecosystem
Result: a restored-equilibrium PE_x or *null*

```
1  begin
2      V_done, a set of evaluated process identifiers, initially empty
3      p_var, the selected variant of a redesigned process, initially null
4      p_m, the process to be changed, initially null
5      PE_x ← PE_p; p_var ← p;
6      V_done ← V_done ∪ {identifier of p};
7      p_m ← GetNextProcess(PE_x, V_done);
8      while p_m ≠ null and p_var ≠ null do
9          p_var ← ProcessChangeForMinConflicts(p_m, PE_x, V_done);
10         if p_var ≠ null then
11             replace p_m in PE_x by p_var;
12             V_done ← V_done ∪ {identifier of p_m};
13             p_m ← GetNextProcess(PE_x, V_done);
14         else
15             PE_x ← null;
16         end
17     end
18     return PE_x;
19 end
```

Empirically, it is shown that ordering variables for value assignment can have substantial impacts on the performance of finding CSP solution [2]. The ordering variables could be either: (i) *static ordering*, the order of variables is defined before the search starts and not be changed until the search complete or (ii) *dynamic ordering*, the next variables to be assigned are dynamically defined at any point depends on the current state of the search. There are some heuristics in selecting variable ordering, i.e. variable with the smallest domain (in dynamic ordering) or variable which participates in the most constraints.

We argue that maintaining the equilibrium in a process ecosystem can be casted as a binary CSP. We can build a constraint network [6], represented in a constraint graph, of the process ecosystems to depict a binary CSP in which each node represents a process model, all possible variants of redesigning a process can be considered as domain value of each node and each edge represents a relationship constraint between processes, as in Figure 3. Process $P1$ in Figure 3a can be mapped into a node $P1$ in Figure 3b, as well as its relationship to process $P2$, i.e. $R1$, which is mapped into an edge between nodes $P1$ and $P2$, and so forth for the remaining processes and relationships. Indeed, the domain value of each node is finite since there exist constraints (at least, end cumulative effects and activity temporal constraints) that must be satisfied by process variants. We consider

Algorithm 2: Generating a restored-equilibrium process ecosystem : *constructive* approach

Input:
p, a changed process model
PE_p, graph of a perturbed-equilibrium process ecosystem
Result: a restored-equilibrium PE_x or *null*

```
1  begin
2      V_done, a set of evaluated process identifiers, initially empty
3      p_var, the selected variant of a redesigned process, initially null
4      p_m, the process to be changed, initially null
5      PE_x ← PE_p; p_var ← p;
6      V_done ← V_done ∪ {identifier of p};
7      p_m ← GetNextProcess(PE_x, V_done);
8      while p_m ≠ null and p_var ≠ null do
9          p_var ← a process variant of p_m which maintains the existing
           relationships with other processes identified in V_done, if no possible
           variant then p_var = null;
10         if p_var ≠ null then
11             replace p_m in PE_x by p_var;
12             V_done ← V_done ∪ {identifier of p_m};
13             p_m ← GetNextProcess(PE_x, V_done);
14         else
15             I_pm ← a path running from p to p_m;
16             p_mdb ← the preceding of p_m in I_pm ;
17             if p_mdb = p then
18                 PE_x ← null;
19             else
20                 remove identifier of p_mdb from V_done;
21                 p_m ← p_mdb;
22                 p_var ← p_mdb;
23             end
24         end
25     end
26     return PE_x;
27 end
```

constraint graph of a process ecosystem as a tuple $G_e = (V, C)$ where V and C denote a set of process nodes and a set of relationship constraints between processes, respectively. In the resulting constraint graph G_e, each process node is of the form $(id, T, E, G, start, end)$ where $id, T, E, G, start, end$ represent ID, set of activities, set of edges, set of gateways, start and end events of a process, respectively. And, each relationship constraint is of the form $(id, source, target, type)$ where $id, source, target, type$ represent ID, source node, target node and type of an inter-process relationship, respectively. An equilibrium in process ecosystem then is considered as a solution in CSP once all constraints are satisfied by value

assigned to each node, i.e. a variant of each process. We might not have a solution for a given perturbed-equilibrium process ecosystem since there does not exist a variant of a particular process node for resolving the violations[5].

Algorithms. We propose two algorithms, i.e. *repair* and *constructive*, for generating a restored-equilibrium process ecosystem, as shown in Algorithms 1 and 2, respectively. The analyst can perform either one or both of them to generate a restored-equilibrium process ecosystem. We implement dynamic ordering in searching process to be evaluated through one which participates in the most constraints, represented by ***GetNextProcess*** function. We search a process in the perturbed-equilibrium process ecosystem which is in the following conditions: (i) not yet evaluated, (ii) violates its relationship constraints with the previously evaluated processes and (iii) participates in the most constraints.

In the repair approach, inspired by Min-Conflicts algorithm [17], we search the new equilibrium of process ecosystems by minimizing conflicts between variants of process being changed with the other processes which are not yet evaluated, and satisfying relationship constraint with the previously evaluated processes. It is represented by ***ProcessChangeForMinConflicts*** function which searches a variant of changed process by satisfying the following criteria: (i) satisfies all relationship constraints with the previously evaluated processes and (ii) has the minimal violations with all the rest processes in the ecosystem. Finally, we would select a variant which has the minimum conflict and continue until all constraints satisfied, shown in Algorithm 1.

In the constructive approach, inspired by Graph-based backjumping algorithm [5], we search a new equilibrium of a process ecosystem by redesigning the process being evaluated to satisfy its constraint with the previous evaluated process until all constraints are satisfied. Once there is no variant of the process being evaluated to satisfy the constraint, we would jump back to the most recent related process (with respect to the process being evaluated) which is already evaluated, as shown in Algorithm 2. Then, this recent process should be redesigned to make the following process to be evaluated satisfy its constraint.

5 Evaluation

We have performed an empirical validation[6] to assess how the repair and constructive approaches perform on different sizes of the process ecosystem. Specifically, we established process ecosystems with sizes ranging from 10 to 80 processes. All these processes have 5-20 activities. We also annotated each activity with immediate effects and had the tool compute the cumulative effects in each process at every stage. The established process ecosystems are equipped with the inter-process relationships discussed earlier. Our framework generates all possible relationships between different processes and presents the constraint graph of every process ecosystem.

[5] Finding an optimal solution would be our next investigation.

[6] All experiments were run on a i3 Intel Core-2.27 GHz, RAM 2.85 Gb laptop.

The experiments are conducted as follows. A process, denoted as $P1$, was selected to have some initial changes, which are then propagated to other related processes to maintain the consistency of the process ecosystem. In general, the total time required to establish a restored-equilibrium process ecosystem is $(s + nt)$ where s is the time our CSP algorithms compute, n is the number of processes needing to be changed (as identified by our tool) and t is the average time for making changes to a process. In our experiments, we assume that when a process is flagged as needing changing, the actual changes would be done by the analyst. As such, we are only interested in the time elapsed for propagating changes in our CSP algorithms.

Table 1. Elapsed time for searching the process ordering and checking the constraints of various sizes of process ecosystems

			Elapsed time	
No. of processes	**No. of violated constraints**	**No. of redesigned processes n**	**Repair mode (sec)**	**Constructive mode (sec)**
10	3	3	64	63
20	8	8	331	329
30	11	11	875	844
40	12	12	1,316	1,272
50	12	12	1,642	1,512
60	13	13	2,199	1,988
70	14	14	2,699	2,443
80	18	18	3,395	3,163

Table 1 describes how the repair and constructive approaches perform in proportion to the size of the process ecosystem in terms of the elapsed time for establishing a new equilibrium of process ecosystem. Our experimental results suggest that the proposed approach is efficient (i.e. helps analysts propagate changes regardless of the complexity of the inter-process relationships in the process ecosystem) and scalable in propagating changes to maintain the equilibrium of a medium-sized process ecosystem (up to 80 processes). Additionally, performing the repair approach to get a restored-equilibrium process ecosystem takes longer time than performing the constructive approach. This could be explained as follows. In the repair approach, we need to verify the consistency between all processes that are related to the process being modified whereas in constructive approach, we only check the consistency between the process being modified and related processes that were already modified. However, we have not taken into account the complexity of redesigning an individual process in our experiments (i.e. how long it would take, for the analyst, to redesign a process that needs changing). Furthermore, we have not analyzed the complexity of our algorithms in order to correlate the elapsed time with parameters represented by the

three leftmost columns of Table 1. The scalability of our framework for dealing with a large-sized ecosystem will be addressed in our future investigations.

6 Related Work

Change propagation approaches have been intensively investigated in software evolution/maintenance, engineering management and software modeling (see, e.g. [1, 3, 4]). Recently, this approach has also been applied to BPM and service computing (see, e.g. [18, 20]). However, there exist little work on change propagation in process model collections [7], as can be seen in [8]. Weidlich *et al.* [20] attempt to determine a change region in another model by exploiting the behavioral profile of corresponding activities due to a model change. Their behavioral profile relies on three relations, which are based on the notion of weak order, between nodes in a process graph. Wang *et al.* [18] present analysis of dependencies between services and their supporting business processes. On the top of this analysis, they define change types and impact patterns which are used to analyze the necessary change propagation occurring in business processes and services. To the best of our understanding, these researches are only dealing with a pair of business artifacts. The closely related work to our proposed framework is done by Ekanayake *et al.* [8], which deals with processes in a collection. They propose change propagation based on the shared fragments between process models. To propagate changes, they develop a special data structure for storing these fragments and process models. Once changes are made to a fragment, all processes which this fragment belongs to are considered to be changed. This fragment-based approach would be closely related to one of our research interests, namely change propagation for the specialization-generalization relationship.

We leverage constraint networks [6] using CSP approach in propagating the changes between processes. To the best of our knowledge, CSP technology has not been used in the existing researches to deal with change propagation in a complex process repository. We are interested in how changes on one process can be properly propagated to the related processes to maintain the consistency-equilibrium of a process ecosystem. We focus on three kind of relationships between semantic effect-annotated BPMN models, i.e. part-whole, specialization-generalization and inter-operation.

7 Conclusion and Future Work

Being inspired by CSP approach, we have proposed a novel framework for managing relationship-preserving change propagation in process ecosystems. This framework can assist the process analysts in maintaining the equilibrium of their process ecosystems within a complex process repository. Future work includes development of techniques for generating process variants for a given process, finding the optimal solution with minimal change strategy of restored-equilibrium process ecosystem and performing experiments on our framework using case-studies taken from the industry.

References

1. Aryani, A., Peake, I.D., Hamilton, M.: Domain-based change propagation analysis: An enterprise system case study. In: 2010 IEEE International Conference on Software Maintenance (ICSM), pp. 1–9. IEEE (2010)
2. Bartak, R.: Constraint propagation and backtracking-based search. Charles Universität, Prag (2005)
3. Chua, D.K.H., Hossain, M.A.: Predicting change propagation and impact on design schedule due to external changes. IEEE Trans. on Eng. Management (99), 1–11
4. Dam, H.K., Winikoff, M.: Supporting change propagation in UML models. In: 2010 IEEE International Conf. on Software Maintenance (ICSM), pp. 1–10. IEEE (2010)
5. Dechter, R.: Enhancement schemes for constraint processing: Backjumping, learning, and cutset decomposition. Artificial Intelligence 41, 271–312 (1990)
6. Dechter, R.: Constraint Processing. Morgan Kaufmann Publishers (2003)
7. Dijkman, R., Rosa, M., Reijers, H.: Managing large collections of business process models-current techniques and challenges. Comp. in Industry 63(2), 91–97 (2012)
8. Ekanayake, C.C., La Rosa, M., ter Hofstede, A.H.M., Fauvet, M.-C.: Fragment-Based Version Management for Repositories of Business Process Models. In: Meersman, R., Dillon, T., Herrero, P., Kumar, A., Reichert, M., Qing, L., Ooi, B.-C., Damiani, E., Schmidt, D.C., White, J., Hauswirth, M., Hitzler, P., Mohania, M. (eds.) OTM 2011, Part I. LNCS, vol. 7044, pp. 20–37. Springer, Heidelberg (2011)
9. Ghose, A., Koliadis, G.: Model eco-systems: preliminary work. In: The Fifth Asia-Pacific Conf. on Conceptual Modelling, pp. 19–26. Australian Comp. Society (2008)
10. Ghose, A., Koliadis, G., Chueng, A.: Rapid Business Process Discovery (*R*-BPD). In: Parent, C., Schewe, K.-D., Storey, V.C., Thalheim, B. (eds.) ER 2007. LNCS, vol. 4801, pp. 391–406. Springer, Heidelberg (2007)
11. Hill, J.B., Cantara, M., Deitert, E., Kerremans, M.: Magic quadrant for business process management suites. Tech. rep., Gartner Research (2007)
12. Hinge, K., Ghose, A., Koliadis, G.: Process SEER: A tool for semantic effect annotation of business process models. In: IEEE International Enterprise Distributed Object Computing Conference, EDOC 2009, pp. 54–63. IEEE (2009)
13. Koliadis, G., Ghose, A.: A Conceptual Framework for Business Process Redesign. In: Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Soffer, P., Ukor, R. (eds.) BPMDS 2009 and EMMSAD 2009. LNBIP, vol. 29, pp. 14–26. Springer, Heidelberg (2009)
14. Koliadis, G., Ghose, A.: Verifying semantic business process models in inter-operation. In: Int. Conf. on Services Computing 2007, pp. 731–738. IEEE (2007)
15. Kurniawan, T.A., Ghose, A.K., Lê, L.-S., Dam, H.K.: On Formalizing Inter-process Relationships. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) BPM Workshops 2011, Part II. LNBIP, vol. 100, pp. 75–86. Springer, Heidelberg (2012)
16. La Rosa, M., Dumas, M., Uba, R., Dijkman, R.: Business process model merging: an approach to business process consolidation. ACM Transactions on Software Engineering and Methodology (TOSEM) (in press, 2012)
17. Minton, S., Johnston, M.D., Philips, A.B., Laird, P.: Minimizing conflicts: A heuristic repair method for constraint satisfaction and scheduling problems. Artificial Intelligence 58, 161–205 (1992)

18. Wang, Y., Yang, J., Zhao, W.: Change impact analysis for service based business processes. In: IEEE International Conference on Service-Oriented Computing and Applications, SOCA 2010, pp. 1–8. IEEE (2010)
19. Weber, B., Rinderle, S., Reichert, M.: Change Patterns and Change Support Features in Process-Aware Information Systems. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007. LNCS, vol. 4495, pp. 574–588. Springer, Heidelberg (2007)
20. Weidlich, M., Weske, M., Mendling, J.: Change propagation in process models using behavioural profiles. In: Int. Conf. on Services Computing, SCC 2009, pp. 33–40. IEEE (2009)
21. White, S.A., Miers, D.: BPMN: Modeling and Reference Guide. Future Strategies Inc. (2008)