Mining version histories for change impact analysis in business process model repositories

Hoa Khanh Dam[a], Aditya Ghose[a]

[a]*School of Computer Science and Software Engineering*
University of Wollongong, Australia.

Abstract

In order to remain competitive and sustainable in today's ever-changing business environments, organizations need to frequently make changes to their business activities and the corresponding business process models. One of the critical issues that an organization faces is change impact analysis: estimating the potential effects of changing a business process to other processes in the organization's business process repository. In this paper, we propose an approach to change impact analysis which mines a version history of a business process model repository. Our approach then identifies business process models that have been co-changed in the past and uses this knowledge to predict the impact of future changes. An empirical validation on a real business process model repository has showed the effectiveness of our approach in predicting impact of a change.

Keywords: Business Process Change Management, Change Impact Analysis

1. Introduction

A business process is defined as consisting of a set of activities, performed by their relevant roles or collaborators, to intentionally achieve a set of common business goals [1]. Business processes are the core assets of any enterprise, covering many aspects in industry such as design, engineering, man-

Email addresses: `hoa@uow.edu.au` (Hoa Khanh Dam), `aditya@uow.edu.au` (Aditya Ghose)

URL: `http://www.uow.edu.au/∼hoa/` (Hoa Khanh Dam), `http://www.uow.edu.au/∼aditya/` (Aditya Ghose)

ufacturing, purchasing, physical distribution, production management and supply chain management. Organizations committed to long-term business process management (BPM) may have repositories of hundreds or even thousands of business process models. For example, the IBM BIT Process Library has 735 process models [2], the SAP Reference Model contains 604 process models [3], and there are 6,000+ process models in Suncorp's process model repository for insurance [4]. On the other hand, the ever-changing business environment (due to various reasons such as new customer requirements, global competition pressures, new regulations, new IT solutions, economic down turn, etc.) demands organizations to constantly consider changing their business activities in order to remain competitive and sustainable in the long term.

Business process models are essential knowledge assets for an organization (with hundreds and thousands of business process models [5]) to manage its business processes in terms of documenting and implementing procedures, control their execution, analyze their performance, and improve them (i.e. business process management [6, 7]). Recent studies (e.g. [8]) have demonstrated various perceived benefits of using business process models as the basis for process improvement, understanding, communication, execution, analysis and simulation. In particular, both the practitioner and vendor groups participating the study conducted in [8] agreed that support for continuous improvement of an organization's business processes (in order to react to changes in its business environment) is the core benefit of business process models. Changes in process models need to be put into practice immediately in order for process improvement to be effective [1]. Recent work (e.g. [9]) provide techniques for automatic execution of business process models or develop process engines that can interpret process models and enact them automatically. The explicit documentation of business processes in those contexts facilitates change management in terms of quickly identifying what needs to be changed and implementing those changes rapidly.

However, making changes to large, complex repositories of business process models is a highly challenging task. This is mainly due to the *ripple effect* caused by a change. Specifically, a change made to one business process can potentially affect a range of other processes that are related to the process being changed. For example, changes initially made to a sub-process (e.g. adding a new activity) may lead to secondary, additional changes made to the processes that contain this sub-process. Such changes made to those processes may lead to further changes in other related processes. In a large repository of hundreds or even thousands of business process models, it becomes critical to determine the impact of a change, which is the core focus of our paper.

Change impact analysis usually starts with the business analyst examining the change request and determining the processes initially affected by the change (i.e. the *primary changes*). The business analyst then determines other process models in the business process repository that are potentially affected and required to be changed. Changes made to those impacted processes may also potentially affect other processes and thus the impact analysis continues this procedure until a complete impact set is obtained. Change impact analysis plays a major part in planning and establishing the feasibility of a change in terms of predicting the cost and complexity of the change (before implementing it). This helps reduce the risks associated with making changes that have unintended, expensive, or even disastrous effects on existing business operations.

Organizations face a range of challenges in managing change in the context of large and complex collections of business processes. While an important class of change management problems pertain to process instances, our focus in this paper is however exclusively on process designs/models. Here, we take a model-based approach to impact analysis which examines impacts to the business processes before the implementation of such changes. An appropriate decision can therefore be made (before any detailed implementation of the change is considered) on whether to implement a specific set of changes based on what business process models are likely to be impacted and thus on the likely change cost. Earlier decision making and change planning are clearly important in the context of rigorous change management. We also acknowledge that there may be a gap between the actual execution of a process and what is being described in its model, and change impact analysis for process instances is an alternative.

Although business process management research is gaining increasing attention from both industry and academia, there has been very little work on supporting change impact analysis in business process model repositories [5]. Some recent work (e.g. [10]) only focus on identifying dependency relationships among different entities in a single process in order to analyze the impact of a change made to one part of the process to other parts of the process. The work in [11] specifically aims to support change impact analysis between services and business processes in a service-oriented environment. The recent work in [12] addresses the issue of propagating changes to maintain consistency within a process repository, which is part of change implementation rather than change impact analysis.

Traditional approaches to change impact analysis in business processes tend to focus on establishing inter-process relationships (i.e. dependencies) and using this knowledge for impact analysis. These approaches rely on a classification of relationships between business processes (e.g. [13, 12]). However, basic dependency-based impact analysis techniques are considered to be conservative in that they consider all possible transitive closure of inter-process relationships. Results produced by those techniques may have enormous impact sets, which are sometimes unnecessary or even too large to be of practical use [14]. In addition, establishing inter-process relationships in such a way that precisely reflect the semantic nature of dependency between processes (e.g. annotating process models with semantic effects as done in [15]) may be labour-intensive and time consuming.

In this paper, we propose an alternative approach which focuses on detecting factual inter-process dependencies as manifested in the evolution of the process models. Our approach mines the revision history of a process model repository, and identifies processes that have been frequently changed at the same time to identify *co-variation patterns* between them. This approach computes the impact based on the heuristic that *processes that have been changed together in the past (co-variation coupled) will be likely changed together in future.* We have performed an empirical validation using a real business process repository to compare the effectiveness of our approach against the basic dependency-based impact analysis technique.

The structure of this paper is as follows. In the next section, we briefly describe how business processes are defined using the standard Business Process Modeling Notation and how they are annotated with semantic effects. In section 3, we present a generic framework for change impact analysis in business process repositories. We then discuss in detail a basic inter-process relationship (dependency-based) analysis approach (section 4) and our revision history mining approach (section 5). We report on our evaluation in section 6 and discuss related work in section 7. We then conclude and outline our future work in section 8.

2. Background

2.1. Business Process Modeling Notation

While there are a range of modeling notations for business processes, for our purposes we use the Business Process Modeling Notation (BPMN) which is a standard for business process modelling [16]. It provides graphical notation for specifying various types of activities, decision responsibilities, control and data flow in business process within one organization and in cross-organizational settings. BPMN has been widely used in the industry due to its powerful notation which is readily understandable by both the business stakeholders and the technical developers. Figure 1 shows a BPMN diagram describing a typical process of assessing insurance claims and a process of making claim payments.

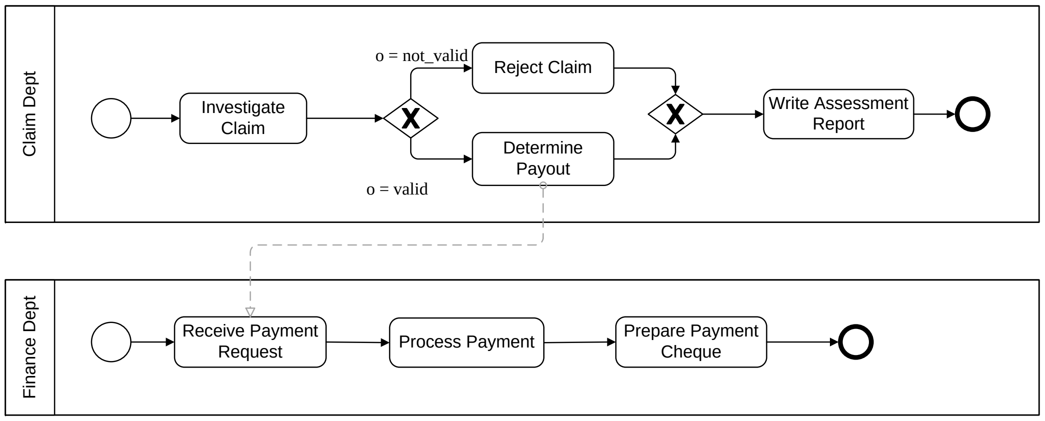

Figure 1: The processes of assessing an insurance claim and making claim payment

There are three major categories of notational elements in BPMN: Flow Objects, Connecting Objects and Swimlanes. The elements known as Flow Objects are: Events, Activities, and Gateways. Activity is the most common type of Flow Objects which describes the kind of work which must be performed. A Task is a atomic activity while a Sub-Process is a composite activity, which contains additional levels of business process detail. In figure 1, “Investigate Claim” and “Reject Claim” are examples of activity tasks. A Gateway is used to control the divergence and convergence of Sequence Flow and determines branching, forking, merging, and joining of paths. There are different types of gateways (exclusive, inclusive, complex, and parallel) and the behavior of each type gateway specifies how many of the gates will be available for the continuation of flow. Figure 1 has an XOR gateway modelling decisions that are based on the validity of a claim.

Flow Objects are connected in three different ways: by sequence flows, message flows or association. Sequence Flows are used to show the order in which a series of flow objects have to be completed. A Message Flow describes the exchange of messages between two process participants while an Association is used to associate information (e.g. an Artifact or text) with Flow Objects. Process participants are represented in a BPMN diagram as a Pool. For instance, there are two pools, i.e. "Claim Dept" and "Finance Dept", in the process in figure 1 and they exchange messages with one another, e.g. the "Determine Payout" task in the "Claim Dept" pool sends a message to the "Finance Dept" pool. In order to separate different processes, a pool can be sub-partitioned into swimlanes. Finally, Artifacts are used to provide additional information about the process. A typical artifact in BPMN is Data Object which represents data required or produced in an activity.

2.2. Semantically effect annotated process models

Precise change impact analysis requires inter-process relationships be formally defined to capture the deeper semantics of the processes. Even if business processes are properly modelled in BPMN, only the coordination semantics of business processes are explicitly described. The semantics of processes in terms of their effects are *not* explicitly described, and thus are *not* interpreted by automated machinery. For example, a BPMN process model might require that task A must precede task B, but does not provide any indication of what is done by tasks A and B (beyond what might be implicit in their nomenclature), i.e. their effects. We are unable to determine from a process design in BPMN what the effects achieved by a process might be at any point in the process design, which is necessary to establish a truly semantical relationship between processes. The problem is not alleviated by taking recourse to the formal semantics of process design notations such as BPMN. Such semantics, as pointed out above, only describe the coordination aspects of a process. The solution we adopt involves explicit semantic annotation of process models by analysts. To ensure practitioner accessibility, and to avoid placing an unduly heavy burden of annotation on analysts, this approach only requires that analysts provide a description of the immediate effects of each process task, i.e., a context-independent specification of the functionality (together with relevant associated ramifications) of each task.

These are then accumulated into cumulative effect annotations in a context-sensitive manner, such that the cumulative effect annotations associated with any task in a BPMN process model would describe the effects achieved by the process were it to execute up to that point.

We acknowledge that there are a number of settings where process tasks in BPMN (and especially in more formal modelling languages) may be linked directly to services which execute the tasks. In such cases, the immediate effects of a process task can be derived from the description of the associated service. Such service specifications may already exist and thus can be used to retrieve semantic effects.

There is a large body of work that explores the use of semantic annotation of business process designs [17, 18, 15, 19, 20, 21, 22, 23, 24]. A large body of work also addresses the problem of semantic annotation of web services in a similar fashion [25, 26, 27, 28, 29]. Common to all of these approaches is the specification of *post-conditions*, which is what we primarily leverage in defining inter-process relationships. For our purposes, two aspects of the post-conditions (or effects) are important. First, post-conditions should be sensitive to process context, i.e., the post-conditions of a task at a certain point in a process design should reflect not just the effects achieved by executing that task but also the accumulated effects of the prior tasks in the process design that have been executed. Second, non-determinism must be accommodated in relation to post-conditions.

A number of the process annotation approaches referred to above achieve contextualization of post-conditions by using a device originally used in AI planning - add-lists and delete-lists of effects (to accumulate effects, the elements of the add-list are added to the prior set of effects, while the elements of the delete-list are removed from the prior set of effects). Others, such as [21, 15], use a state update operator derived from the literature on reasoning about action. Our work is in line with this latter approach - we use a syntactic state update operator based on the Possible Worlds Approach (PWA) [30]. The choice of this particular operator is mainly a matter of convenience, and is adequate for the purposes of establishing the feasibility of this approach, while other operators, such as one based on the Possible Models Approach (used by [21]) could also be used. Our approach also offers an alternative to the approach of translating BPMN models into a Petri net analysis toolset (e.g. [31]) for the purpose of performing semantic analysis.

The need for permitting non-determinism in effects stems from two observations. First, in any process with XOR-branching, one might arrive at a given task via multiple paths, and the contextualized post-conditions achieved must be contingent on the path taken. Since this analysis is done at design time, we need to admit non-deterministic effects since the specific path taken can only be determined at run-time. Second, many state update operators, such as the one associated with the PWA, generate non-deterministic outcomes, since inconsistencies (that commonly appear in state update) can be resolved in multiple different ways. Our approach assumes that each task/activity is annotated with post-conditions (in the formalization below, we shall assume them to be unique, as much of the literature does, but this can be easily generalized to admit non-deterministic post-conditions), which are contextualized via a process of *effect accumulation*. We shall assume that all tasks (and their post-conditions) are drawn from an *enterprise capability library*. In this approach, we are able to answer, for any point in a process design, the following question: what will have happened if the process executes up to this point? The answer is a mutually exclusive set of *effect scenarios*, any one of which might describe the actual state of affairs at that point in the execution of the process design. In the following, we will outline the effect accumulation machinery, in terms of how state update occurs over contiguous tasks and across XOR- and AND-split and merge gateways. We note that events are to be annotated in exactly the same way as activities, but omit these from the formalization in the interests of simplicity and brevity. We also remind the reader that the approach outlined below, represents one of many valid approaches to the semantic annotation of business process designs. The main arguments of this paper hold independent of which particular scheme for effect annotation is adopted.

Contiguous Tasks: We define *pair-wise effect accumulation*, which, given an ordered pair of tasks with effect annotations, determines the cumulative effect after both tasks have been executed in contiguous sequence. We assume that the effect annotations have been represented in conjunctive normal form (CNF) where each clause is also a *prime implicate* (this provides a non-redundant canonical form). Simple techniques exist for translating arbitrary sentences into the conjunctive normal form, and for obtaining the prime implicates of a theory (references ommitted for brevity). Let $\langle t_i, t_j \rangle$ be an ordered pair of tasks connected via a sequence flow such that t_i precedes t_j, let e_i be an effect scenario associated with t_i and e_j be the post-conditions associated with t_j. Let $e_i = \{c_{i1}, c_{i2}, \ldots, c_{im}\}$ and $e_j = \{c_{j1}, c_{j2}, \ldots, c_{jn}\}$ be sets of clauses (we can view CNF sentences as sets of clauses, without loss of generality). If $e_i \cup e_j$ is consistent, then the resulting cumulative effect, denoted by $acc(e_i, e_j)$, is $e_i \cup e_j$. Else, we define $acc(e_i, e_j) = e_j \cup X$ where $X \in \{C | C \subseteq e_i, C \cup e_j$ is satisfiable and there exists no C'such that $C \subset C' \subseteq e_i$ and $C' \cup e_j$ is satisfiable$\}$. In other words, the cumulative effect of the two tasks consists of the effects of the second task plus as many of the effects of the first task as can be consistently included. Alternative resolutions of inconsistencies are thus an additional source of non-determinism in the design-time description of the effects of a process. Effects are only accumulated within participant lanes. Activities which are not connected to any preceding activity via a control flow link are annotated with the cumulative effect $\{e\}$ where e is the immediate effect of the task in question. We accumulate effects through a left-to-right pass of a participant lane, applying the pair-wise effect accumulation procedure on contiguous pairs of tasks connected via control flow links. The process continues without modification over splits.

XOR- and AND-splits: Consider an n-way XOR-split. The role of guard conditions in semantic annotation is important. Consider the first activity t on an outgoing sequence flow from an OR- or XOR-split. Let E be the set of effect scenarios annotating the activity immediately preceding the XOR-split and let $E' \subseteq E$ be the maximal subset of E' in which each effect scenario is consistent with the guard condition c associated with that outgoing flow. Then the set of effect scenarios of t is given by $\{acc(e \wedge c, e_t) \mid e \in E'\}$, where e_t is the post-condition of t and $e \wedge c$ is assumed without loss of generality to be represented as a set of prima implicates. In other words, we enforce consistency of effect scenarios with guard conditions and propagate these guard conditions through models as if these were themselves effects. Note that we could require that only effect scenarios that entail a guard condition get propagated along a given branch, but do not, because this would involve making unrealistic assumptions about the completeness of annotations provided by analysts. For a pair of activities located on either side of an AND-split, we perform pairwise effect accumulation as if these were contiguous tasks.

More specifically, we deal with AND-split as follows. Let t be the task immediately preceding an AND-split and let $\{t_1, t_2, ..., t_n\}$ be the set of tasks immediately following an AND-split (thus t_1 is the first task on the first outgoing flow from the AND-split, t_2 is the first task on the second outgoing flow from the AND-slit, and so on). Then the set of effect scenarios associated with t_i (for $i = 1, 2, ..., n$) will be given by $acc(e_t, e_{t_i})$ where e_t is an effect scenario associated with task t and e_{t_i} represents the post-condition of t_i. We make a simplified assumption that a well-formed (and correctly constructed) process model will not involve interacting effects (i.e. state transitions on the same set of objects) on multiple parallel flows. This permits us to accumulate effects over each flow without reference to the effects being obtained over other flows parallel to the current one. Somewhat more complex machinery exists for checking that this assumption does indeed hold for a given process model but that discussion is tangential to the key objectives of this paper.

AND-merge: Let t_1 and t_2 be the two tasks immediately preceding an AND-join. Let the associated effect scenarios with these tasks be $E_1 = \{es_{11}, es_{12}, \ldots, es_{1m}\}$ and $E_2 = \{es_{21}, es_{22}, \ldots, es_{2n}\}$ respectively. Let e be the post-condition, and E the set of associated effect scenarios for a task t immediately following the AND-join. We define $E = \{acc(es_{1i}, e) \cup acc(es_{2j}, e) | es_{1i} \in E_1 \text{ and } es_{2j} \in E_2\}$. Note that we do not consider the possibility of a pair of effect scenarios es_{1i} and es_{2j} being inconsistent, since this would only happen in the case of intrinsically and obviously erroneously constructed process models. In general, a correctly modeled process would not involve the same business object(s) being impacted on parallel flows. However, it is possible to modify our approach so that we replace every set of parallel flows with sequences representing all possible interleavings of the activities in the parallel flows, and taking the set of effect scenarios after the AND-merge to be the union of the set of effect scenarios obtained from effect accumulation over each such sequence. This approach would lead to a blowout in the number of possibilities to be considered, but might be useful in situations where guarantees of correctly constructed parallel flows are not available.

XOR-merge: Consider the same setting as with AND-merge above, except with t_1 and t_2 being separated by an XOR-merge gateway. Let e be the immediate effect annotation, and E the cumulative effect annotation of a task t immediately following the XOR-merge. We define $E = \{acc(es_i, e) | es_i \in E_1 \text{ or } es_i \in E_2\}$.Note that these definitions for AND-merge and XOR-merge generalize directly to cases with more than 2 incoming flows.

We note that it is impossible to deal with loops (in this approach or any other) if the number of iterations is not pre-determined at run-time, but we can perform approximate checking by partial loop unraveling.

It is also important to note that we pay the costs associated with recording all possible effect scenarios. The alternative is to record only the minimal set of effects that can be guaranteed at any point in a process design. This approach is computationally significantly more efficient to ours, and can be of considerable value in providing support in process design editors. This distinction between recording all potential effects versus recording the effects that can be guaranteed to hold parallels the distinction between the *skeptical* and *credulous* approaches to non-monotonic reasoning. In a manner akin to the discourse in non-monotonic reasoning there are no absolute arguments as to which approach is the best - the nature of the application determines which approach is best suited. In our setting, we need to make reference to complete effect scenarios in the inter-process relationships that we aim to analyze. The cost in computational complexity is tolerable, since the analysis is offline (unlike settings, for example, where we might seek to provide real-time advice to process designers).

Finally, it is important to note that this approach permits multiple occurrences of what is essentially the same functionality, simply by generating a new task/activity ID for each distinct occurrence (the first time T is executed, we call it T-1, the second time, T-2 and so on). This is also a common device used to get around a similar constraint imposed by some process modeling notations (that a given task/activity can only appear once in a process design). Indeed, it makes sense to adopt the position that the second execution of a task is actually semantically distinct from the first application of that same task and so on. In our approach to effect accumulation, the set of accumulated effect scenarios associated with each distinct execution of the same functionality will necessarily be distinct. Note also our earlier comments about the difficulty associated with dealing with loops in design time (not just in our approach, but any other).

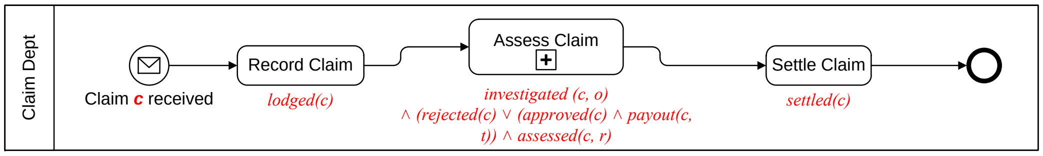
Claim Dept
Claim c received
Record Claim
lodged(c)
Assess Claim
investigated (c, o) ∧ (rejected(c) ∨ (approved(c) ∧ payout(c, t)) ∧ assessed(c, r)
Settle Claim
settled(c)

Figure 2: The semantic annotated process model for handling an insurance claim (P_1)

Figure 2 shows a process of handling an insurance claim (P_1) where each activity in this process is annotated with an immediate effect which is expressed, in the interests of simplicity, in predicate logic (although these annotations could be in more expressive notations). For example, the immediate effect of activity "Record Claim" is $lodged(c)$ where c denotes an insurance claim. This annotation allows us to determine, at design time, the *cumula-*

tive effects of a process P if it were to be executed up to a certain activity a in the business process model (denoted as $CE(P, a)$) by accumulating effects through a left-to-right pass (i.e. stepping through each flow in the process model, from start to end) of a participant pool[1]. For example, the cumulative effects CE of process P_1 in Figure 2 up to activity “Assess Claim” (AC) is $CE(P_1, AC) = \{lodged(c) \wedge investigated(c, o) \wedge (rejected(c) \vee (approved(c) \wedge payout(c, t)) \wedge assessed(c, r)\}$, which is the accumulation of the immediate effects of the first and the second activity in process P_1. Note that the cumulative effects at a particular point are represented by a set of alternative *effect scenarios*, each of which corresponds to an alternative path (due to exclusive gateways for example) in the process model or to a particular resolution of inconsistencies during state update (given that such inconsistencies might be resolved in multiple possible ways, as in the case of the multiple extension problem [32]). For example, the *end cumulative effects* (i.e. the end outcome) of process P_2 in Figure 4 are a set of two effect scenarios $es1$ and $es2$, i.e. $CE(P_2) = \{es1, es2\}$ where:

- $es1 = investigated(c, o) \wedge rejected(c) \wedge assessed(c, r)$ (corresponding to the path “Investigate Claim” - “Reject Claim” - “Write Assessment Report”), and
- $es2 = investigated(c, o) \wedge approved(c) \wedge payout(c, t) \wedge assessed(c, r)$ (corresponding to the path “Investigate Claim” - “Determine Payout” - “Write Assessment Report”).

The cumulative effects $CE(P_2) = \{es1, es2\}$ can also be viewed as $es1 \vee es2$, i.e. $(investigated(c, o) \wedge rejected(c) \wedge assessed(c, r)) \vee (investigated(c, o) \wedge approved(c) \wedge payout(c, t) \wedge assessed(c, r))$, which is equivalent to $investigated(c, o) \wedge (rejected(c) \vee (approved(c) \wedge payout(c, t)) \wedge assessed(c, r)$. Therefore, $CE(P_2)$ can also be written in a combined form (i.e. a singleton set) as $CE(P_2) = \{investigated(c, o) \wedge (rejected(c) \vee (approved(c) \wedge payout(c, t)) \wedge assessed(c, r)\}$.

3. Change impact analysis framework

In this section, we describe a framework for change impact analysis in a process repository (see Figure 3). Impact analysis starts with the business

[1]We refer to [15] for a detailed description of the effect accumulation.

analyst examining the change request, identifying the processes in the current process repository (i.e. $PR_{current}$ in Figure 3) initially affected by the change, and possibly making changes to those processes. Such *primary changes* yield a new version $PR_{modified}$ of the process repository. Our framework then compares the two versions and automatically detects the set of processes that have been changed by the primary changes. Processes in this set are called *initially changed processes*. The version differencing relies on a taxonomy of changes on a process repository, which in its simplest form consists of three main possible changes: adding a new process, deleting a process and modifying an existing process.

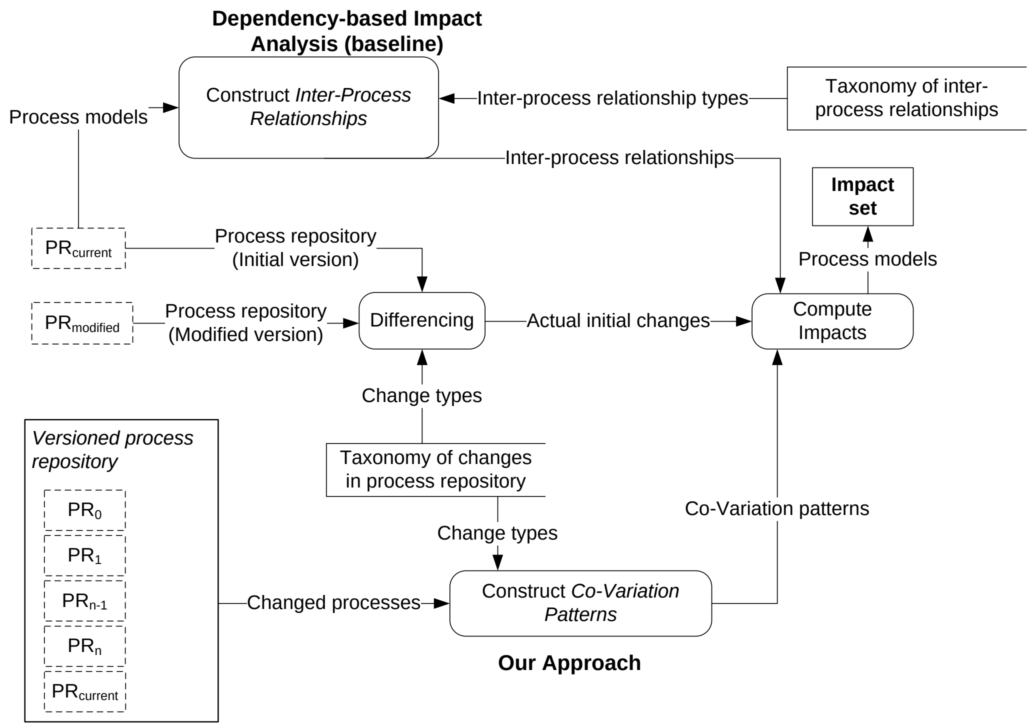
Dependency-based Impact Analysis (baseline)
Construct Inter-Process Relationships
Inter-process relationship types
Taxonomy of inter-process relationships
Process models
Inter-process relationships
Impact set
PRcurrent
Process repository (Initial version)
Process models
PRmodified
Process repository (Modified version)
Differencing
Actual initial changes
Compute Impacts
Change types
Versioned process repository
Taxonomy of changes in process repository
PR0
PR1
PRn-1
PRn
PRcurrent
Change types
Co-Variation patterns
Changed processes
Construct Co-Variation Patterns
Our Approach

Figure 3: A change impact analysis framework for process repositories

We then need to calculate the impact of the primary changes to the process repository: the set of processes impacted by the primary changes. Process P_i is said to be impacted by a change made to process P_j if this change potentially requires a modification of P_i (which is referred to as the *impacted process*). Our approach to change impact analysis relies on mining the evolution history of processes in a process repository to predict the impact of a change to the current process repository. This approach examines the version history of the process repository (versions PE_0, PE_1, ..., $PE_{current}$) to identify processes that are changed at the same time (i.e. in the same revision) and form co-variation patterns among them. This approach works under the assumption that business processes that were changed together in the past will be likely changed together again in future.

We will compare the prediction performance of our approach against a basic depenendency-based impact anlaysis tehcnique which acts as a baseline and relies on examining the inter-process relationships existing in a process repository. This baseline technique calculates the impact set based on the heuristics that if one process is related to the other, then the later may require modification if the former has been changed. Therefore, this technique traverses the inter-process relationship graph to identify other processes that are related to the initial changed processes, and form an impact set. Those impacted processes also relate to other processes, and thus the impact analysis continues this traversal until a complete transitive closure graph is obtained.

4. Dependency-based impact analysis

This approach focuses on identifying relationships between processes in a process repository. The semantic effects can be used to establish semantic relationships between processes. For this baseline approach, we used four types of inter-process relationships: part-whole, inter-operation, generalization-specialization [12], and data-dependent. We briefly describe them as below.

- Part-whole: this type of relationship exists between two processes when one process is required by the other to fulfill some of its functionalities. Specifically, the "whole process" must have an activity that represents the functionalities (in terms of cumulative effects) of the "part" process, which is commonly named a sub-process. Formally, process P is part of process Q iff there exists an activity a in P such that $CE(P, a) = CE(P \uparrow^a Q, a)$ where $CE(P, a)$ is the cumulative effects of executing process P up to activity a, and $P \uparrow^a Q$ is the process model obtained by viewing Q as the sub-process expansion of activity a in P. For example, the expansion of process P_2 in the sub-process activity "Assess Claim" (AC) results in process $P_1 \uparrow^{AC} P_2$, and $CE(P_1 \uparrow^{AC} P_2, AC) = \{lodged(c) \wedge investigated(c, o) \wedge (rejected(c) \vee (approved(c) \wedge payout(c, t)) \wedge assessed(c, r)\}$. Therefore,

$CE(P_1, AC) = CE(P_1 \uparrow^{AC} P_2, AC)$, and thus P_2 is a part of P_1 in the part-whole relationship between the two processes.

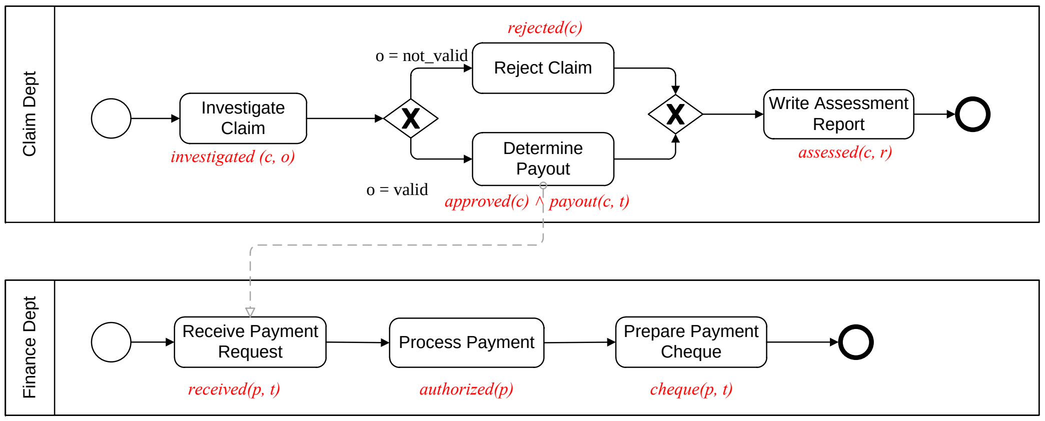

Figure 4: The semantic annotated process models for assessing an insurance claim (P_2) and making claim payment (P_3)

- Inter-operation: this type of dependency exists between two processes when there is at least one message exchanged between them and there is no cumulative effect contradiction between activities that involve in the message exchanging. Formally, process P and Q has an inter-operation relationship iff both of the following holds: (i) there exists a message flow between activity p in P and activity q in Q; and (ii) $CE(P, p) \cup CE(Q, q) \nvdash \bot$. Note that in formal logic $\bot$ (falsum) represents a contradiction, and $\varphi \nvdash \bot$ indicates that φ is not a contradiction. For example, there is a message flow between activity "Determine Payout" (DP) in process P_2 and activity "Receive Payment Request" (RP) in process P_3 and there is no contradiction between $CE(P_2, DP)$ and $CE(P_3, RP)$, and thus there exists an inter-operation relationship between the two processes in Figure 4.

- Generalization-specialization: this type of dependency covers the situation when one process (i.e. the specialization) is a functional extension of the other (i.e. the generalization). The specialized process not only inherits the same functionalities from its generalization but also has additional functionalities. This can be realized by having some additional activities or enriching the immediate effects of the existing activities.

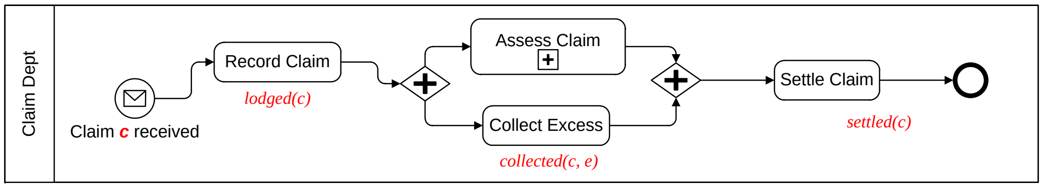

Figure 5: The process of handling claim with excess payment (P_4)

Both methods extend the indented end cumulative effects of the specialized process. Formally, process Q is a specialization of process P iff both of the following holds: (i) $\forall es_p \in CE(P) \cdot \exists es_q \in CE(Q) \cdot es_q \models es_p$; and (ii) $\forall es_q \in CE(Q) \cdot \exists es_p \in CE(P) \cdot es_q \models es_p$. Note that $x \models y$ means x semantically entails y. For example, the end cumulative effects of the handling claim process (P_4) in Figure 5 is the end cumulative effects of process P_1 (Figure 2) plus $collected(c, e)$ which is the effect of collecting the excess e associated with claim c. As a result, there is a generalization-specialization relationship between P_1 and P_4 where the former is a generalization of the latter.

- Data-dependent: Any data-dependent relationship between two process models can be reduced to a setting where one of the processes relies on (and potentially waits for) data to be generated by another process. It is interesting to note that this is a special case of the inter-operation relationship where one process waits on a message (or data item) to be generated by another process. More specifically, this can be considered as an informational inter-operation dependency, but in this case we also need to check message semantics to ensure that there is no cumulative effect contradiction between activities that involve in the message exchanging and the message itself. Formally, process P and Q has an data-dependent relationship iff both of the following holds: (i) there exists a message flow m between activity p in P and activity q in Q; and (ii) $CE(P,p) \cup CE(Q,q) \cup E(m) \nvDash \perp$ where $E(m)$ is the set of effects associated with message m.

The impact set is computed by calculating the transitive closure of all inter-process relationships existing in a process repository. For example, assume that we would like to compute the impact set of changing P_1. Due to the part-whole relationship existing between P_1 and P_2 and the generalization-specialization existing between P_1 and P_4, both P_2 and P_4 are added to the impact set of P_1. In addition, since P_2 is potentially impacted and P_2 has an inter-operation relationship with P_3, process P_3 is also potentially impacted and should be included in the impact set. Note that the impact set is computed by identifying both direct (e.g. process A is related to process B) and indirect (e.g. A is related to B, B related to C and thus A related to C) inter-process relationships.

5. Mining revision history approach

The above approach requires all business process models in a process repository to be fully annotated with semantic effects. To do so, each activity in each process model needs to be annotated with immediate effects. This is highly time-consuming and expensive, especially with large process repositories of hundreds or even thousands of processes. For example, the IBM BIT Process Library [2] contains 735 process models, each of which has on average 27 activities, which requires 19,845 annotation tasks to be manually done.

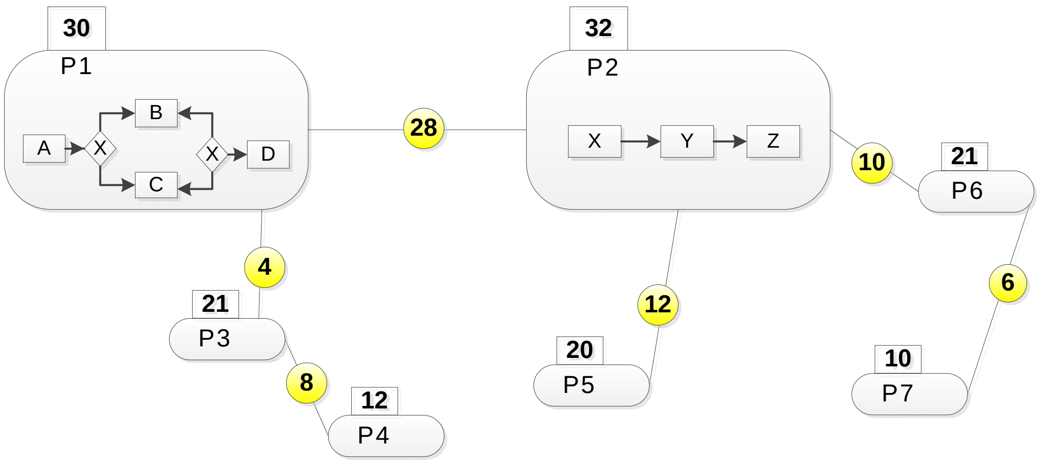

Figure 6: An example of co-variation patterns between business processes

Hence, we propose an alternative approach to change impact analysis that is orthogonal to inter-process relationship analysis. More specifically, our approach leverages information that is available from mining revision histories of a process repository. The revision history of a business process repository contains important information about how and why the processes evolved over time. The revision history can reveal which parts of the process repositories are coupled by co-variation: "when process P_i was changed, process P_j was also modified too", which we refer to as *co-variation patterns*.

Figure 6 shows an example of co-variation patterns within a process repository. Each process is annotated with the number of changes throughout its history – for instance, process P1 has been changed 30 times whilst process P2 has been changed 32 times. Both processes have been changed together 28 times, indicating that there exists a co-variation pattern between them. Each co-variation pattern is associated with a *strength* level, which is the number of times the two processes have been changed together. For example, the co-variation pattern between processes P1 and P2 is very strong since P1 has been changed 30 times, 28 of which was with P2 being changed at the same time. On the other hand, the co-variation pattern between processes P1 and P3 is not strong since they have been changed together only 4 times, out of 30 times in which P1 has been changed (and 21 times for P3).

We define a history of a process repository as a series of revisions wherein each revision there is a set of changes simultaneously submitted to the repository by a business analyst. Our technique retrieves such changes from existing version archives provided by any process repositories that support version controlling. For process repositories that support the traditional version controlling (as done in CVS or Subversion), the changed processes which were submitted in the same commit can be considered as co-changed (similarly to changed files submitted in the same commit in CVS or Subversion). In this case, we then examined the commit logs to identify co-changed processes. In the lack of such version controlling feature, we may follow the classical sliding window principle: if two processes were changed in the period of time (e.g. same day), they are considered to be co-changed. The sliding window time is user-definable and domain specific. Alternatively, in a different domain, we could examine other documentation to identify co-changed processes. For example, we could examine the release logs to identify processes that have been changed in the same release, or in the case of the processes in RosettaNet used in our evaluation (see Section 6), we used the version of DTD and XML Schema as the reference for identifying processes changed to conform with a particular version of DTD or XML Schema.

A co-variation pattern (p_i, p_j) exists between two processes p_i and p_j if they have been changed in the same revision. We define the *support count* $supp(p_i, p_j)$ for a co-variation pattern (p_i, p_j) as being the number of times (frequency) p_i and p_j changed together in the history of the process repository containing them. The support count indicates how much evidence is available to justify the co-variation coupling between processes p_i and p_j. For example, the support count for the co-variation pattern between processes P_1 and P_2 in Figure 6 is 28, and between P_2 and P_5 is 3, whereas between P_4 and P_5 is 0. Table 1 shows the support counts for all possible co-variation patterns between the processes in Figure 6. Zero entries indicate that there is no co-variation pattern existing between the two processes.

Process	P1	P2	P3	P4	P5	P6	P7
P1	30	28	4	0	0	0	0
P2	28	32	0	0	12	10	0
P3	4	0	21	8	0	0	0
P4	0	0	8	12	0	0	0
P5	0	12	0	0	20	0	5
P6	0	10	0	0	0	21	6
P7	0	0	0	0	5	6	10

Table 1: Support count

Process	P1	P2	P3	P4	P5	P6	P7
P1	1	28/30	4/30	0	0	0	0
P2	28/32	1	0	0	12/32	10/32	0
P3	4 /21	0	1	8/21	0	0	0
P4	0	0	8/12	1	0	0	0
P5	0	12/20	0	0	1	0	0
P6	0	10/21	0	0	0	1	6/21
P7	0	0	0	0	0	6/10	1

Table 2: Confidence

We have also another metric to measure the strength of a co-variation pattern: of all changes to a process, how often it changed together with another process. For this means, we define the *confidence* $conf(p_i, p_j)$ for a co-variation pattern (p_i, p_j) from the perspective of process p_i is $conf(p_i, p_j) = \frac{supp(p_i, p_j)}{|p_i|}$ where $|p_i|$ is the number of times process p_i have changed in the history. Similarly, the confidence from the perspective of process p_j is $conf(p_j, p_i) = \frac{supp(p_i, p_j)}{|p_j|}$. For example, the confidence for the co-variation pattern between processes P_1 and P_2 from the perspective of P_1 in Figure 6 is 28/30 (i.e. 0.93), indicating this is a strong co-variation coupling. By contrast, $conf(P_1, P_3)$ is 4/30 (i.e. 0.13), indicating the co-variation pattern between processes P_1 and P_3 is weak. Table 2 summarizes the confidence for all possible co-variation patterns between the processes in Figure 6. Note that the confidence is not symmetric (as in the support count table).

Algorithm 1: CalculateTotalImpacts(): Calculate total impacts of a change using the historical approach

```
Input:
$CS$, the set of atomic changes initially made
$TblSupport$, the support count table
$TblConfidence$, the confidence table
$T_{supp}$, the support threshold
$T_{conf}$, the confidence threshold
Output: $IPS$, the set of impacted processes in the process repository
begin
    foreach change $c_i$ in CS do
        Let $p_i$ be the process changed by $c_i$
        $IPS \overset{\text{def}}{=} IPS \cup \{p_i\}$
        foreach entry ($p_i$, $p_j$) in $TblSupport$ and $i \neq j$ do
            if supp($p_i$, $p_j$) $\geq T_{supp}$ then
                foreach entry ($p_i$, $p_j$) in $TblConfidence$ do
                    if conf($p_i$, $p_j$) $\geq T_{conf}$ then
                        $IPS \overset{\text{def}}{=} IPS \cup \{p_j\}$
                    end
                end
            end
        end
    end
end
```

In this revision history mining approach, we need to construct the support count table and the confidence table for all processes in a process repository based on its revision history. This approach also requires as input a set of initially changed processes and two *user-provided thresholds*: T_{supp} for the support threshold, and T_{conf} for the confidence threshold. These thresholds are to eliminate the cases where processes are changed together coincidentally. Algorithm 1 describes how the impact set is computed. For each process p_i that was initially changed, we scan through the row for p_i in the support table and retrieve entries with the support count greater or equal to T_{supp}. We then look for those entries in the confidence table and retrieve only the entries with the confidence greater or equal to T_{conf}. We then add the processes associated with those entries to the impact set.

For example, assuming that we need to find out the impact of process P_2 being changed in our earlier example in Figure 6. We further assume that the support threshold is 2 and the confidence threshold is 0.4. We first go through row P_2 in the support count table (Table 1) and retrieve 3 entries with support count greater than or equal 2: (P_2, P_1), (P_2, P_5) and (P_2, P_6). We then look up the confidence table (Table 2) for the confidence values of those entries: $conf(P_2, P_1) = 0.88$, $conf(P_2, P_5) = 0.38$ and $conf(P_2, P_6) = 0.31$. Since the confidence threshold is 0.4, only process P_1 is added to the impact set of P_2.

6. Evaluation

An important question that we would like to address in this evaluation is: *how well the mining revision history approach works, compared with the traditional, basic inter-process relationship approach.* We would like to understand whether the mining revision history approach can offer similar accurate and precise impact analysis to the inter-process relationship approach, but without the expensive cost of annotating business process models. We now discuss how we have designed such an experiment, the measures that we used, the outcomes and some major threats to the validity of our experiment.

6.1. Measures

Given a set of processes initially changed PI (which is the input to both of our impact analysis approaches), the effectiveness measurement we used involves two sets: the set of *potentially* impacted processes which are estimated by an impact analysis technique (i.e. the estimated set E) and the set of processes *actually* affected/changed (i.e. the actual set A).

Given a set of processes initially changed PI, the aim of an impact analysis technique is determining the portion of the process repository truly affected by the change (set A). However, an impact analysis technique may

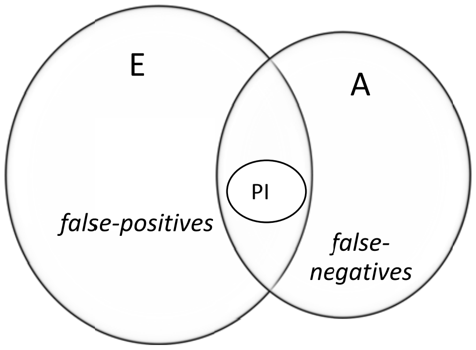

Figure 7: The impact of primary changes PI (E is the estimated set and A is the actual set)

not be fully accurate in identifying the impact set. Figure 7 illustrates this issue. Overestimating impact generates *false-positives* (i.e. processes that are in E but not in A), which forces the business analysts to spend additional, unneeded time investigating the impact set that contain unnecessary information. On the other hand, underestimating impact produces *false-negatives* (i.e. processes that are in A but not in E), which leads the business analysts to omit important impacts of a change. If such impacts continue being omitted when implementing the change, they may cause inconsistencies in a process repository, which may result in more serious issues in the organization's operation.

We use two relative measures: *precision* and *recall*, which are associated with false-positives and false negatives. Precision and recall are the most widely-used metrics in the information retrieval literature and have been recently adapted to the impact analysis setting (e.g. [33]). Precision is defined as the ratio between the correctly predicted (i.e. the intersection of E and A) and the total of predicted processes (i.e. E): $Precision = \frac{|E \cap A|}{|E|}$. Recall is the ratio between correctly predicted processes (i.e. the intersection of E and A) and the total of actually affected processes (i.e. A): $Recall = \frac{|E \cap A|}{|A|}$. On the one hand, a perfect precision score of 1.0 means that every process predicted as being impacted by an impact analysis technique was actually changed. On the other hand, a perfect recall score of 1.0 means that all changed processes were predicted as being impacted by the technique. Our objective is to achieve high precision and high recall values, meaning that we aim to recommend all (recall of 1) and only expected processes (precision of

1).

6.2. Baseline

Ideally, the evaluation could be done by given the tool (that implements both of our approaches) to a group of selected users, who would be asked to work with the tool. We would then collect and analyze data such as their feedback, the change outcomes, etc. to assess how much assistance the tool provides them. Unfortunately, such a comprehensive user study is time consuming and would require an organization to adopt our tool for a period of time. Instead, we propose to use the historical data stored in a real process repository to compare the performance of the two techniques based on our earlier definitions of precision and recall.

For our evaluation, we have used the repository of RosettaNet Standards' Partner Interface Processes (PIP)[2]. RosettaNet Standards provide frameworks that allow the interoperability of business processes across the global supply chain. RosettaNet specifications include the business process definitions and technical elements for interoperability and communication. A RosettaNet PIP defines business processes between trading partners. There are 132 PIPs provided with RosettaNet Standards. Those business processes represent the backbone of the trading network such as distribution of product information, order management, inventory management, marketing information management, service and support, manufacturing, administrative functionalities, and partner product and service review. Those PIPs also represent cross-enterprise processes which involve more than one type of trading partners.

We have evaluated the basic dependency-based impact analysis approach (described in Section 4), and used its results as a baseline for our evaluation. To do so, we have manually annotated the RosettaNet PIPs by going through RossettaNet documentation, in particular the Partner Interface Process (PIP) specifications, and identifying the effects of each activity in a PIP. Both semantic annotation and the baseline evaluation were done before we analyzed the version history of PIPs (see the next section). This is to avoid any influence on our choices of annotating effects or establishing inter-process relationships. Since version 02.07.00 released on 09 April 2009 provides full documentation of all the Partner Interface Processes in RossettaNet, we an-

[2]The full archive of RosettaNet PIPs is available at http://www.rosettanet.org

notate all the PIPs provided and described in this release and used the next release as the test version. The semantic annotation enables us to establish that there are in total 81 direct inter-process relationships identified in this version of RosettaNet PIPs which comprises of 34 inter-operation, 14 part-whole, 24 data-dependent and 9 generalization-specialization relationships.

We then computed the set of *all* processes (denoting as Δ) that were *actually* changed in the test version. If changes made to a process p (primary changes) in Δ caused the modification of other processes (secondary changes), then those affected processes must be part of Δ. Hence, for each process p in Δ we did the following:

- Create a test case with process p being the process initially changed (primary change) and $A = \Delta - \{p\}$ being the set of processes that are actually changed (i.e. the expected outcome).

- With p as the input, we compute the set of processes E which our approach estimates, and compare it with the expected outcome A using precision and recall measures.

We summarized all the precision-recall pairs into a single precision-recall pair by taking the mean value of these pairs. The mean recall and precision values recorded for the baseline dependency-based impact analysis are 0.16 and 0.12 respectively.

6.3. Mining approach

In order to evaluate the predictability of our mining approach, we analyzed the full archive of RosettaNet Standards' Partner Interface Processes (PIP). There were in total 37 revisions that we were able to extract from the archive of Rosettanet PIPs between 10 May 2010 and 6 February 2013, and we used all of them in our evaluation. These are obtained from the 397 Partner Interface Process (PIPs) specifications (including processes and their versions) listed in PIP Directory on the Rosettanet website (refer to Appendix A for more detail). We extracted revision information from the process name. For example, *"0A1_NotificationOfFailure_V02_00_00"* indicates a version of process $0A1$ namely *Notification Of Failure* which conforms to the DTD or XML Schema version *V02_00_00*. We assume that processes which conform to the same version of DTD or XML Schema were simultaneously changed since the change in the DTD or XML Schema may have required changing those processes (refer to Appendix A for a list of versions). For example, we inferred that processes *"0A1_NotificationOfFailure_V02_00_00"*, *"3A1_RequestQuote_V02_00_00"*, and *"2A1_DistributeProductCatalogInformation_V02_00_00"* were changed together since they tied to the same version *V02_00_00* of the DTD.

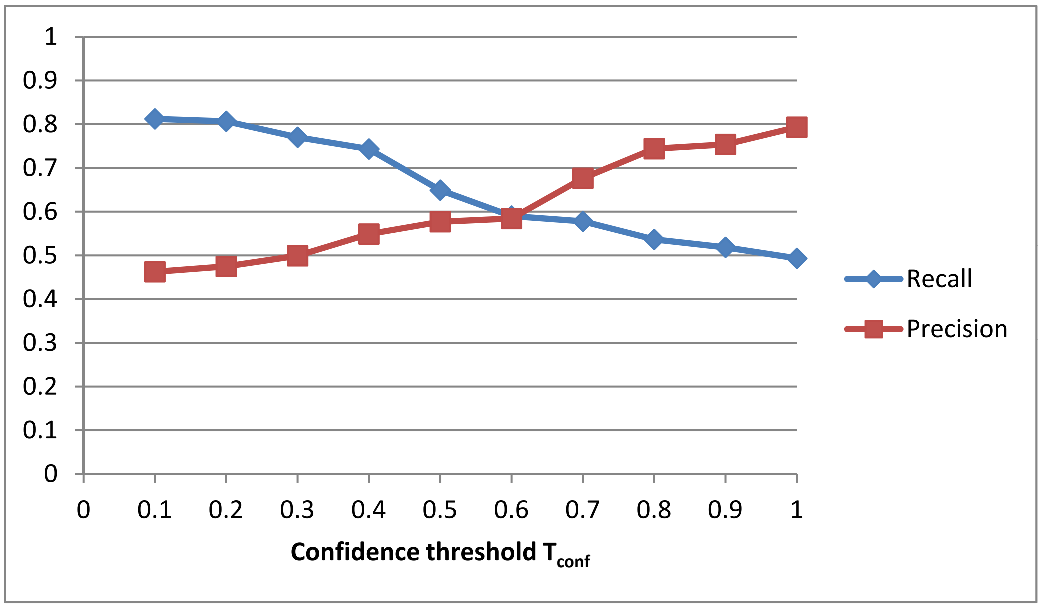

Figure 8: Recall and precision values for support threshold $T_{supp} = 1$

We performed the evaluation using K-fold cross validation and thus divided these revisions into a *training set* and a *test set*. More specifically, since there are only 37 revisions in our data set, we have decided to use the *leave-one-out cross-validation* in which each test set contains only 1 revision and the remaining 36 revisions were used for training. This procedure was repeated 37 times so that each revision is used for testing exactly once. This approach has the advantage of utilizing as much available data as possible for training. In addition, the test sets are mutually exclusive and they effectively cover the entire data set we used for our evaluation. The test set was derived from the test revision by establishing the set of processes Δ that were changed in this revision. We then followed the same procedure described in the previous section to compute the mean precision and recall.

Since the thresholds for support count and confidence may affect the performance of our mining approach, we varied them and performed separate

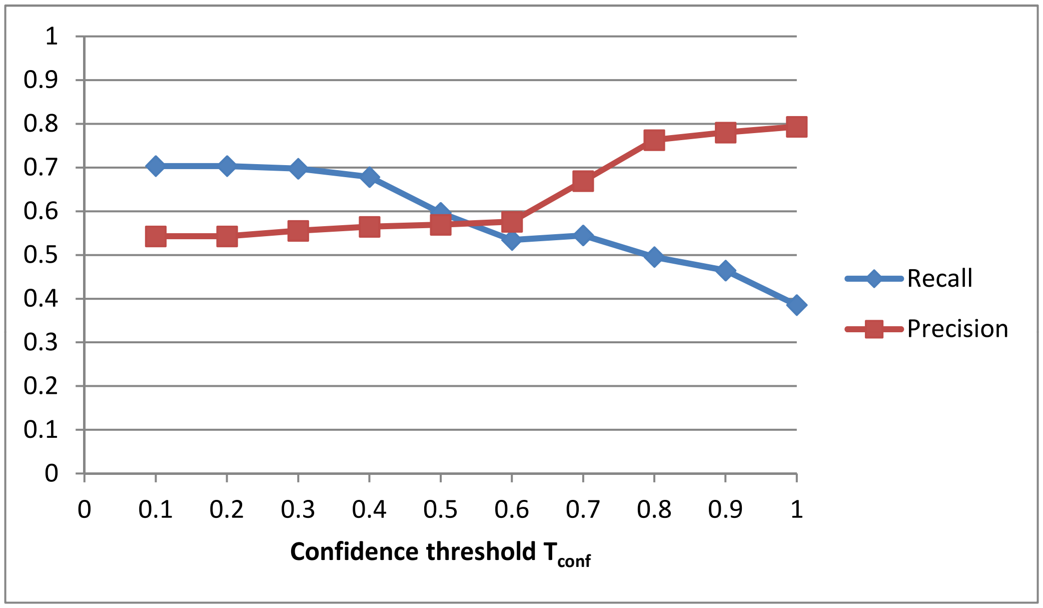

Figure 9: Recall and precision values for support threshold $T_{supp} = 2$

experiments[3] for different combinations of the support count and confidence thresholds. Specifically, we investigated 10 different values for the confidence threshold: 1, 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, and 0.1, and investigated 3 different values for the support threshold: 1, 2, and 3.

Figures 8, 9 and 10 shows the recall and precision values for the RosettaNet PIPs with the support threshold of 1, 2 and 3 respectively and varying confidence thresholds from 0.1 to 1. For example, in Figure 8 our mining approach achieves for a support of 1 and confidence of 0.1 a recall of 0.81 and a precision of 0.46, meaning that:

- The recall of 0.81 indicates that our mining technique's prediction correctly included 81% of all process that were actually changed in the given revision.
- The precision of 0.46 indicates that 46% of all processes that are predicted by our mining technique were correct.

[3]All experiments reported in this paper were performed on a PC running Windows 7 and Java v1.7.0_11, with a Intel Core i5-2500 3.30GHz CPU and 8GB RAM.

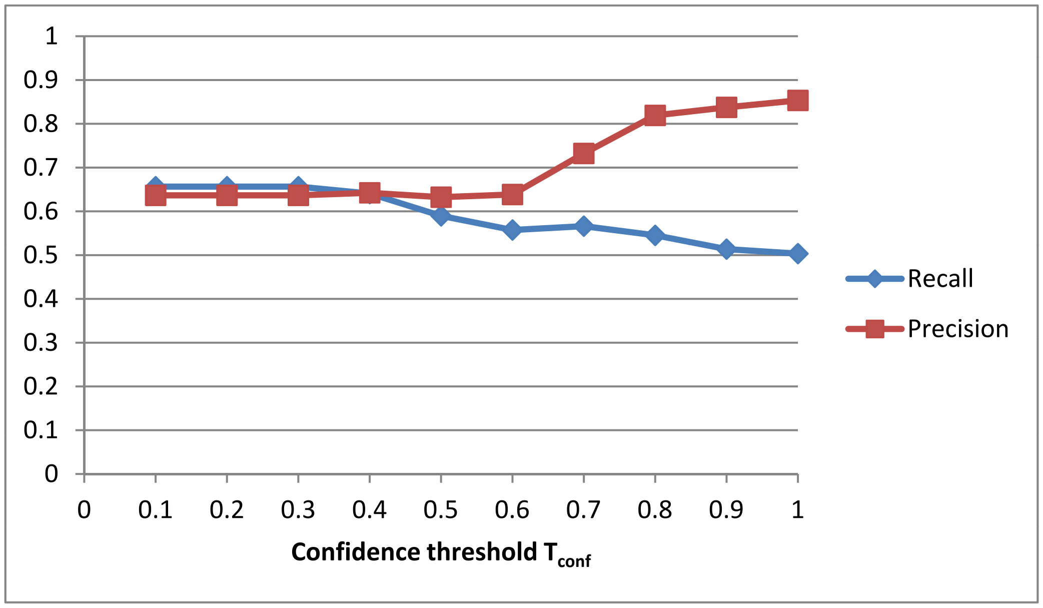

Figure 10: Recall and precision values for support threshold $T_{supp} = 3$

Figures 8, 9 and 10 also show that increasing the confidence threshold generally leads to increase the precision, but decrease in the recall as our mining technique becomes more cautious. A similar observation can be seen in increasing the support threshold. In addition, as can be seen in Figures 8, 9 and 10, high support and confidence thresholds are required for high precision. However, such values lead to a low recall, which indicates a trade-off between precision and recall. In practice, graphs presented in Figures 8, 9 and 10 are useful for choosing suitable support and confidence thresholds for a specific business process repository. The results have also demonstrated our mining approach significantly outperforms the baseline dependency-based impact analysis in both recall and precision. It is also useful to note that the recall (86%) and precision (46% – in many cases it is above 70% as in Figures 8, 9, and 10) are generally better than most of the other work in change impact analysis. For example, the seminal work [34] on guiding software changes achieved a recall of only 26% and precision of 15%.

We also note that the low precision and recall in some cases may be due to the limited size of the version archive that our approach used to learn co-variation patterns. Learning from larger archives (e.g. consisting of hundreds or thousands of versions of process models) which cover better the historical evolution of processes may improve the predictability of our mining approach. We would however emphasize that securing those large, industrial archives of business processes is difficult and is part of our future work (once they become available to us).

6.4. Threats to validity

We have investigated 37 revisions of 132 business processes in RosettaNet PIPs. Since the process repository and its revision histories chosen for our evaluation are limited to RosettaNet PIPs, we acknowledge that they would not be representative for all kinds of process model repositories. Therefore, as part of future work we would aim to apply our techniques to revision histories of other business process models. In practice, if changes are enforced in a strict change management, which would result in changes that are more logically related, our approach would give a higher precision and recall (i.e. a better predicability). Finally, usefulness for the user can only be determined by studies with humans, which is also part of our future work.

7. Related work

In the past few years, dealing with large repositories of business models has attracted an increasing attention from both industry and academia since it has become more common see organizations dealing with repositories of hundreds or thousands business process models [5]. Various techniques which address different aspects of managing process model repositories have been proposed in literature such as querying for a particular process model (e.g. [35]), searching for similar process models (e.g. [36]), mining for business rules from process model repository [37] and identifying clones or variants in process models (e.g. [38]).

To the best of our knowledge, there has been however no work in identifying the ripple impact of a change made to one process to other processes in a large process model repository. Nonetheless, some recent work have addressed the change propagation (a closely related problem of change impact analysis) issue: given a set of primary changes made to a process, what additional changes are needed to other processes in a process repository. The work in [12] has addressed this issue based on the conjecture that given a suitable set of inter-process relationships (which are formally defined through through their annotated semantic effects), change propagation can be done by finding and fixing inconsistent relationships in a process repository, which are caused by primary changes made to the repository. The underlying change propagation mechanism of this framework is leveraged upon the well-known Constraint Satisfaction Problem (CSP) technology. By contrast, the work in [39] propagates changes based on a pre-defined propagation policy for each process model. In addition, the work in [39] proposes to capture relationships between process models in terms of the common parts (process fragments) that they share. Their approach automatically decomposes a process model into a tree of single entry and single exit (SESE) fragments. A change made to a process model fragment may have an impact on all process models which contain this fragment. This allows the users to analyze an impact of a proposed change on process models. However, there are other inter-process relationships that are not covered in their approach. For example, a dependency may exist between two processes that interact with each other (message exchange) or one process relies on (and potentially waits for) data to be generated by another process. A pair of process models related via a generalization/specialization may also not have fragments in common. Such processes do not necessarily share common fragments but changing one of them may affect the others. Our approach is thus distinct to that in [39], and potentially more generally as well.

Some recent work has also addressed the issue of change impact analysis for business processes. The work in [10] defines a number of dependency types revolving around entities of a process model (e.g. routing dependency, data dependency, and role dependency) and uses them for analyzing the impact of a change from one entity to other entities in the same process. Some work have also addressed the issue of how changing a business process affects the services it supports. For example, the work in [11] defines a number of change impact patterns which capture the effect of some specific changes (e.g. the insertion of an activity in a process requires a certain operations be added to the corresponding services).

The work of business process architecture [40, 41] is related to the inter-process relationships we presented in this paper. In a broad sense, business process architecture captures the relations between processes within a process collection. The inter-process relationships that we have adopted here can fit in a process architecture. On the other hand, relationships that have been proposed in the business process architecture literature such as composition, specialization, trigger and information flow are similar to what we have used here for the basic dependency-based analysis technique. Business process architectures can range from minimal architectures which simply define the relationships between key business processes to more detailed architectures which also define links between processes and strategies, policies and resources. An elaborate process architecture can serve as a valuable senior management tool but it needs to be up-to-date to correctly reflect the actual linkages among architectural elements [42]. An up-to-date business process architecture allows us to quickly define the impact of proposed changes. For example, if a well-defined architecture defines the relationships between processes and subprocesses, and between processes and IT resources and training resources, one can quickly estimate how a specific business process change affects IT or training. Therefore, an organization would benefit from the creation and constant maintenance of a business process architecture in terms of improving its ability to deal with change in a rapid and efficient manner. In practice, it is however challenging to create and more importantly update a large detailed business process architecture [42], and thus our mining approach can serve as an alternative. Our mining approach is not tied to any specific inter-process relationship types or process architectures.

There have been a proliferation of techniques (see a recent survey in [14]) proposing to support change impact analysis of procedural, object-oriented systems or agent-oriented systems (seminal work presented in [43] or more recent work such as [44, 45, 46, 33, 47, 48]), which can be classified into two groups: static and dynamic analysis. Static impact analysis techniques (e.g. [49] for object-oriented systems or a number of techniques reported in [43]) usually perform either program slicing or graph traversals (by analysing the source code) to compute impact sets. These techniques are considered to be conservative in that they consider all possible program inputs and behaviours. Results produced by static analysis may have enormous impact sets, which are sometimes unnecessary or even too large to be of practical use. Recently, the work in [46] proposed a variable granularity approach to improve the precision of an impact analysis by allowing the software engineer to choose a level of granularity (e.g. classes, or class members or code fragments) during an iterative process, i.e. the software engineer interactively works with the tool in each step to correct the mistakes (i.e. false positives) made by the tool. Recent techniques (e.g. [50, 51, 52, 34]) leverage the emerging mining software repositories technology to make use of the historical co-changes to compute the impact of changes in source code.

8. Conclusions and future work

In this paper, we have described two *distinct* approaches to change impact analysis in business process models. The basic dependency-based approach requires semantic annotation of business process model but does *not* require revision histories. On the other hand, the mining approach (which is our main contribution) does *not* require semantic annotation. Our mining approach uses the revision history of a process repository to predict change impact based on the assumption that processes that were changed frequently in the past will be likely changed together again in future. We have performed an experiment on 37 revisions in the archive of 132 business processes in RosettaNet's PIPs, and the results have shown that our approach can achieve highly precise prediction of change impact. The results from our evaluation have also suggested that the mining version history approach is more accurate than a basic dependency-based analysis which relies on computing the transitive closure of inter-process relationships. The mining approach however requires the availability of revision histories. In practice, this revision histories might not always be available. The creation of a revision history of the process repository can take time before the input is useful to perform a change impact analysis. However, version management and configuration management are supported by many of today's business process model repositories [53]. For example, the repository for integrated process management [54], Fragmento [55], and Apromore [56] provide support for allowing allowing multiple versions of a process to be created and enabling users to maintain relations between versions of processes. Hence, version histories can be found in those process model repositories.

Future work involves assessing our techniques with large industrial business process model repositories and implementing our technique into tooling support for business analysts and managers. An important class of change management problems pertain to process instances. Hence, an important direction of future research is exploring change impact analysis at the process instance level. This approach can leverage process mining techniques [57] to extract relationships between processes based on process execution behaviour (e.g. process event logs) and use such inter-process relationships for impact analysis. The advantages of this approach is that the dependencies between processes can be identified without the need of semantic annotation or version histories. Furthermore, this might reveal dependencies that were not intended by design and that the process designer might not have been aware of (i.e. the behaviour extracted from the logs may be non-conformant to that described in the process models). In addition, extracted actual dependencies between processes may give more accurate prediction of change impact since they are derived at the process execution level. However, if a process design has never (or rarely) been enacted (so that the available logs are limited or non-existent), dependencies between this process and other processes may not be accurately identified. In addition, since the granularity of our current change impact analysis is at the process level (rather at the finer-grain of activities and tasks), we believe that doing this at process models is sufficient.

References

[1] H. Smith, P. Fingar, Business Process Management: The Third Wave, Meghan-Kiffer Press, 2006.

[2] D. Fahland, C. Favre, B. Jobstmann, J. Koehler, N. Lohmann, H. Völzer, K. Wolf, Instantaneous soundness checking of industrial business process models, in: Proceedings of the 7th International Conference on Business Process Management, BPM '09, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 278–293.

[3] T. Curran, G. Keller, A. Ladd, SAP R/3 business blueprint: understanding the business process reference model, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1998.

[4] M. La Rosa, M. Dumas, R. Uba, R. Dijkman, Business process model merging: An approach to business process consolidation, ACM Trans. Softw. Eng. Methodol. 22 (2) (2013) 11:1–11:42.

[5] R. Dijkman, M. La Rosa, H. A. Reijers, Managing large collections of business process models - current techniques and challenges, Computers in Industry 63 (2) (2012) 91–97.

[6] W. M. P. van der Aalst, Business process management: A comprehensive survey, ISRN Software Engineering 2013 (2013) 1–37.

[7] M. Weske, Business Process Management: Concepts, Languages, Architectures, 2nd Edition, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2012.

[8] M. Indulska, P. Green, J. Recker, M. Rosemann, Business process modeling: Perceived benefits, in: Proceedings of the 28th International Conference on Conceptual Modeling, ER '09, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 458–471.

[9] J. Fabra, V. De Castro, P. Álvarez, E. Marcos, Controversy corner: Automatic execution of business process models: Exploiting the benefits of model-driven engineering approaches, J. Syst. Softw. 85 (3) (2012) 607–625.

[10] W. Dai, H. D. Covvey, P. S. C. Alencar, D. D. Cowan, Lightweight query-based analysis of workflow process dependencies, Journal of Systems and Software 82 (6) (2009) 915–931.

[11] Y. Wang, J. Yang, W. Zhao, A change analysis tool for service-based business processes, in: Proceedings of the 12th International Conference on Web Information System Engineering, WISE'11, 2011, pp. 336–337.

[12] T. A. Kurniawan, A. K. Ghose, H. K. Dam, L.-S. Lê, Relationship-preserving change propagation in process ecosystems, in: Proceedings of the 10th International Conference on Service-Oriented Computing, ICSOC'12, 2012, pp. 63–78.

[13] T. A. Kurniawan, A. K. Ghose, L.-S. Lê, H. K. Dam, On formalizing inter-process relationships, in: F. Daniel, K. Barkaoui, S. Dustdar (Eds.), Business Process Management Workshops (co-located with the 9th International Conference on Business Process Management), Vol. 100 of Lecture Notes in Business Information Processing, Springer, 2011, pp. 75–86.

[14] S. Lehnert, A taxonomy for software change impact analysis, in: Proceedings of the 12th International Workshop on Principles of Software Evolution and the 7th annual ERCIM Workshop on Software Evolution, IWPSE-EVOL '11, 2011, pp. 41–50.

[15] K. Hinge, A. Ghose, G. Koliadis, Process seer: a tool for semantic effect annotation of business process models, in: Proceedings of the 13th IEEE International Conference on Enterprise Distributed Object Computing, EDOC'09, 2009, pp. 49–58.

[16] Object Management Group, Business Process Model and Notation (BPMN), v1.2, `http://www.omg.org/spec/BPMN/1.2` (2009).

[17] D. Fensel, F. Facca, E. Simperl, Web Service Modeling Ontology, in: Semantic Web Services, 2011, pp. 107–129. doi:10.1007/978-3-642-19193-0.

[18] M. Hepp, F. Leymann, J. Domingue, A. Wahler, D. Fensel, Semantic business process management: a vision towards using semantic Web services for business process management, in: IEEE International Conference on e-Business Engineering (ICEBE'05), IEEE, 2005, pp. 535–540. doi:10.1109/ICEBE.2005.110.

[19] I. Di Pietro, F. Pagliarecci, L. Spalazzi, Model checking semantically annotated services, IEEE Transactions on Software Engineering 38 (2012) 592–608. doi:10.1109/TSE.2011.10.

[20] F. Smith, M. Proietti, Rule-Based Behavioral Reasoning on Semantic Business Processes, in: Proceedings of the 5th International Conference on Agents and Artificial Intelligence, SciTePress, 2013, pp. 130–143.

[21] I. Weber, J. Hoffmann, J. Mendling, Beyond soundness: On the verification of semantic business process models, Distributed and Parallel Databases 27 (2010) 271–343. doi:10.1007/s10619-010-7060-9.

[22] C. Di Francescomarino, C. Ghidini, M. Rospocher, L. Serafini, P. Tonella, Semantcally-aided business process modeling, in: International Semantic Web Conference, 2009, pp. 114–129.

[23] C. Di Francescomarino, P. Tonella, Supporting ontology-based semantic annotation of business processes with automated suggestions, in: Lecture Notes in Business Information Processing, Vol. 29 LNBIP, 2009, pp. 211–223. doi:10.1007/978-3-642-01862-6_18.

[24] A. Ghose, G. Koliadis, Auditing Business Process Compliance, in: Proceedings of the International Conference on Service-Oriented Computing (ICSOC-2007), 2007, pp. 169–180. doi:10.1007/978-3-540-74974-5.

[25] Q. Liang, X. Wu, E. K. Park, T. M. Khoshgoftaar, C. H. Chi, Ontology-based business process customization for composite web services, IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans 41 (2011) 717–729. doi:10.1109/TSMCA.2011.2132710.

[26] D. Martin, M. Paolucci, S. McIlraith, M. Burstein, D. McDermott, D. McGuinness, B. Parsia, T. Payne, M. Sabou, M. Solanki, Naveen Srinivasan, K. Sycara, Bringing semantics to web services: The OWL-S approach, in: Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition, SWSWPC 2004, Vol. 3387, 2005, pp. 26–42.

[27] H. Meyer, On the Semantics of Service Compositions, in: Web Reasoning and Rule Systems, 2007, pp. 31–42.

[28] M. Montali, M. Pesic, W. M. P. van der Aalst, F. Chesani, P. Mello, S. Storari, Declarative specification and verification of service choreographiess, ACM Transactions on the Web 4 (2010) 1–62. doi:10.1145/1658373.1658376.

[29] F. Smith, D. Bianchini, Semi-automatic pocess composition via semantics-enabled sub-processs selection and ranking, in: Enterprise Interoperability V: Shaping Enterprise Interoperability in the Future Internet, Proceedings of the I-ESA Conferences, 2012, pp. 177–187.

[30] M. L. Ginsberg, D. E. Smith, Reasoning about action i: A possible worlds approach, Artificial Intelligence 35 (1987) 233–258.

[31] R. Dijkman, M. Dumas, C. Ouyang, Semantics and analysis of business process models in bpmn, Inf. Softw. Technol. 50 (12) (2008) 1281–1294.

[32] S. Hanks, D. McDermott, Nonmonotonic logic and temporal projection, Artificial Intelligence 33 (3) (1987) 379–412.

[33] M. C. O. Maia, R. A. Bittencourt, J. C. A. de Figueiredo, D. D. S. Guerrero, The hybrid technique for object-oriented software change impact analysis, in: Proceedings of the 14th European Conference on Software Maintenance and Reengineering, IEEE Computer Society, 2010, pp. 252–255.

[34] T. Zimmermann, P. Weissgerber, S. Diehl, A. Zeller, Mining version histories to guide software changes, IEEE Transactions on Software Engineering 31 (6) (2005) 429–445.

[35] T. Jin, J. Wang, M. La Rosa, A. H. M. ter Hofstede, L. Wen, Efficient querying of large process model repositories, Computers in Industry 64 (1) (2013) 41–49.

[36] M. Kunze, M. Weidlich, M. Weske, Behavioral similarity: a proper metric, in: Proceedings of the 9th International Conference on Business Process Management, BPM'11, 2011, pp. 166–181.

[37] J. Polpinij, A. K. Ghose, H. K. Dam, Business rules discovery from process design repositories, in: Proceedings of the 6th IEEE Congress on Services, SERVICES '10, IEEE Computer Society, Washington, DC, USA, 2010, pp. 614–620.

[38] C. C. Ekanayake, M. Dumas, L. García-Bañuelos, M. La Rosa, A. H. M. ter Hofstede, Approximate clone detection in repositories of business process models, in: Proceedings of the 10th International Conference on Business Process Management, BPM'12, 2012, pp. 302–318.

[39] C. C. Ekanayake, M. La Rosa, A. H. M. Ter Hofstede, M.-C. Fauvet, Fragment-based version management for repositories of business process models, in: Proceedings of Cooperative Information Systems, OTM'11, 2011, pp. 20–37.

[40] R.-H. Eid-Sabbagh, R. Dijkman, M. Weske, Business process architecture: Use and correctness, in: Proceedings of the 10th International Conference on Business Process Management, BPM'12, Springer-Verlag, Berlin, Heidelberg, 2012, pp. 65–81.

[41] R. Dijkman, I. Vanderfeesten, H. A. Reijers, Business process architectures: overview, comparison and framework, Enterprise Information Systems, DOI: 10.1080/17517575.2014.928951.

[42] P. Harmon, Business Process Change: A Guide for Business Managers and BPM and Six Sigma Professionals, 2nd Edition, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007.

[43] R. Arnold, S. Bohner, Software Change Impact Analysis, IEEE Computer Society Press, 1996.

[44] T. Apiwattanapong, A. Orso, M. J. Harrold, Efficient and precise dynamic impact analysis using execute-after sequences, in: ICSE

'05: Proceedings of the 27th International Conference on Software Engineering, ACM, New York, NY, USA, 2005, pp. 432–441. doi:http://doi.acm.org/10.1145/1062455.1062534.

[45] J. Law, G. Rothermel, Whole program path-based dynamic impact analysis, in: ICSE '03: Proceedings of the 25th International Conference on Software Engineering, IEEE Computer Society, Washington, DC, USA, 2003, pp. 308–318.

[46] M. Petrenko, V. Rajlich, Variable granularity for improving precision of impact analysis, in: The 17th IEEE International Conference on Program Comprehension, ICPC 2009, Vancouver, British Columbia, Canada, May 17-19, 2009, IEEE Computer Society, 2009, pp. 10–19.

[47] H. K. Dam, A. Ghose, Automated change impact analysis for agent systems, in: Proceedings of the 27th IEEE International Conference on Software Maintenance, ICSM '11, IEEE, Washington, DC, USA, 2011, pp. 33–42.

[48] H. K. Dam, A. Ghose, Supporting change impact analysis for intelligent agent systems, Science of Computer Programming 78 (9) (2013) 1728 – 1750.

[49] L. Li, J. Offutt, Algorithmic analysis of the impacts of changes to object-oriented software, in: Proceedings of the International Conference on Software Maintenance (ICSM' 96), IEEE, 1996, pp. 171–184.

[50] A. E. Hassan, R. C. Holt, Predicting change propagation in software systems, in: ICSM '04: Proceedings of the 20th IEEE International Conference on Software Maintenance, IEEE Computer Society, Washington, DC, USA, 2004, pp. 284–293.

[51] H. Malik, A. E. Hassan, Supporting software evolution using adaptive change propagation, in: ICSM '08: Proceedings of the 24th IEEE International Conference on Software Maintenance, Beijing, China, 2008.

[52] H. Kagdi, J. I. Maletic, B. Sharif, Mining software repositories for traceability links, in: Proceedings of the 15th IEEE International Conference on Program Comprehension, ICPC '07, IEEE Computer Society, Washington, DC, USA, 2007, pp. 145–154. doi:10.1109/ICPC.2007.28.
URL `http://dx.doi.org/10.1109/ICPC.2007.28`

[53] Z. Yan, R. Dijkman, P. Grefen, Business process model repositories - framework and survey, Information and Software Technology 54 (4) (2012) 380–395.

[54] I. Choi, H. Jeong, M. Song, Y. U. Ryu, Ipm-epdl: An xml-based executable process definition language, Comput. Ind. 56 (1) (2005) 85–104.

[55] D. Schumm, D. Karastoyanova, F. Leymann, S. Strauch, Fragmento: Advanced Process Fragment Library, in: Proceedings of the 19th International Conference on Information Systems Development (ISD'10), Prague, Czech Republic, August 25 - 27, 2010, Springer, 2010, pp. 659–670.

[56] M. La Rosa, H. A. Reijers, W. M. P. Van Der Aalst, R. Dijkman, J. Mendling, M. Dumas, L. GarcíA-BañUelos, Apromore: An advanced process model repository, Expert Systems with Applications 38 (6) (2011) 7029–7040.

[57] W. M. P. van der Aalst, Process Mining: Discovery, Conformance and Enhancement of Business Processes, 1st Edition, Springer Publishing Company, Incorporated, 2011.

Appendix A. Version Histories of RosettaNet Partner Interface Processes

We provide here a list of 132 Partner Interface Processes (PIPs) (listed as their ID and name in Table A.3) used in our evaluation and the revisions in which the processes were considered to be changed together (Table A.4).

Table A.3: Partner Interface Processes

ID	Process
0A1	Notification of Failure
0C1	Asynchronous Test Notification
0C2	Asynchronous Test Request / Confirmation
0C3	Synchronous Test Notification
0C4	Synchronous Test Query / Response
1A1	Request Account Setup
1A2	Maintain Account
2A1	Distribute New Product Information
2A3	Query Marketing Information
2A4	Query Sales Promotion & Rebate Information
2A6	Query Product Lifecycle Information
2A7	Query Product Discontinuation Information
2A9	Query Technical Product Information
2A10	Distribute Design Engineering Information
2A12	Distribute Product Master
2A13	Distribute Material Composition Information
2A15	Request Material Composition Information
2A16	Distribute Engineering Information Inquiry
2A17	Notify of Certificate of Analysis
2A18	Notify of Certificate of Analysis Response
2B1	Change Basic Product Information
2B2	Change Marketing Information
2B3	Change Sales Promotion & Rebate Information
2B4	Change Product Technical Information
2B5	Change Product Lifecycle Information
2B7	Notify of Product Change
2B8	Notify of Product Change Query
2B9	Notify of Product Change Acceptance

2C1	Distribute Engineering Change Status
2C2	Request Engineering Change
2C3	Distribute Engineering Change Response
2C4	Request Engineering Change Approval
2C5	Notify of Engineering Change Order
2C6	Notify of Engineering Change Implementation Plan
2C7	Request Bill Of Material
2C8	Notify of Bill Of Material
2C9	Request Approved Manufacturer List
2C10	Notify of Approved Manufacturer List
3A1	Request Quote
3A2	Request Price and Availability
3A3	Request Shopping Cart Transfer
3A4	Request Purchase Order
3A5	Query Order Status
3A6	Distribute Order Status
3A7	Notify of Purchase Order Update
3A8	Request Purchase Order Change
3A9	Request Purchase Order Cancellation
3A10	Notify of Quote Acknowledgment
3A13	Notify of Purchase Order Information
3A14	Distribute Planned Order
3A15	Notify of Quote Request
3A16	Notify of Quote Confirmation
3A17	Distribute Price And Availability Request
3A18	Distribute Price And Availability Response
3A19	Notify of Purchase Order Request
3A20	Notify of Purchase Order Confirmation
3A21	Notify of Purchase Order Change Request
3A22	Notify of Purchase Order Change Confirmation
3A23	Notify of Purchase Order Cancellation Request
3A24	Notify of Purchase Order Cancellation Confirmation
3B1	Distribute Transportation Projection
3B2	Notify of Advance Shipment
3B3	Distribute Shipment Status
3B4	Query Shipment Status

3B5	Request Shipment Change
3B6	Notify of Shipments Tendered
3B11	Notify Of Shipping Order
3B12	Request Shipping Order
3B13	Notify of Shipping Order Confirmation
3B14	Request Shipping Order Cancellation
3B18	Notify of Shipment Documentation
3B19	Notify of Shipping Order Request
3B20	Notify of Shipping Order Confirmation
3B21	Notify of Shipping Order Cancellation Request
3B22	Notify of Shipping Order Cancellation Confirmation
3C1	Return Product
3C2	Request Financing Approval
3C3	Notify of Invoice
3C4	Notify of Invoice Reject
3C5	Notify of Billing Statement
3C6	Notify of Remittance Advice
3C7	Notify of Self-Billing Invoice
3C8	Notify of Return Product Request
3C9	Notify of Return Product Confirmation
4A1	Notify of Strategic Forecast
4A2	Notify of Embedded Release Forecast
4A3	Notify of Threshold Release Forecast
4A4	Notify of Planning Release Forecast
4A5	Notify of Forecast Reply
4A6	Notify of Forecasting Exception
4B2	Notify of Shipment Receipt
4B3	Notify of Consumption
4C1	Distribute Inventory Report
4D1	Notify of Material Release
4E1	Notify of Sales Report
4E2	Notify of Sales Report Acknowledgement
5C1	Distribute Product List
5C2	Request Design Registration
5C3	Create Design Win
5C4	Distribute Registration Status

5C5	Query Registration Status
5C6	Notify of Design Registration Request
5C7	Notify of Design Registration Confirmation
5C8	Notify of Win Claim Request
5C9	Notify of Win Claim Confirmation
5D1	Request Ship from Stock and Debit Authorization
5D2	Notify of Blanket Ship from Stock and Debit Authorization
5D3	Distribute Open Ship from Stock and Debit Authorization Status
5D4	Query Ship from Stock and Debit Authorization Status
5D5	Create Ship from Stock and Debit Claim Status
5D6	Notify of Ship from Stock and Debit Claim Status
5D7	Notify of Ship From Stock And Debit Authorization Request
5D8	Notify of Ship From Stock And Debit Authorization Confirmation
5D9	Notify of Ship From Stock And Debit Claim Request
5D10	Notify of Ship From Stock And Debit Claim Confirmation
6A1	Notify of Service Contract Request
6A2	Notify of Service Contract Reply
6C1	Query Service Entitlement
6C2	Request Warranty Claim
6C3	Notify of Case Request
6C4	Notify of Case Confirmation
6C5	Distribute Service Entitlement Query
6C6	Distribute Service Entitlement Status Response
6C7	Notify of Warranty Claim Request
6C8	Notify of Warranty Claim Confirmation
7B1	Distribute Work in Process
7B2	Query Work in Process
7B5	Notify Of Manufacturing Work Order
7B6	Notify of Manufacturing Work Order Reply
7C6	Distribute Product Quality Event Data
7C7	Notify of Semiconductor Test Data
7C8	Notify of Semiconductor Process Data

Table A.4: Version archive of PIPs

Version	**Processes** *(by ID)*
01_01_01	2A9
01_05_00	3A8, 5C4
11_13_00	3A4
01_01_00	0C1, 0C2, 0C3, 0C4, 2A12, 2A9, 3A8, 3A9, 3B12, 3B13, 3B14, 3B18, 3B19, 3B2, 3B20, 3C1, 3C3, 3C4, 3C6, 3C7, 4B2, 4B3, 5C2, 5C3, 5C4, 5D1, 5D3, 5D5, 6C1, 6C2, 7B1, 7B5, 7C6
11_02_00	2A13, 3A8, 3B12, 3B14, 3B2, 3C3, 3C4, 3C6, 3C7, 4C1, 7C7
01_02_00	0C1, 0C2, 0C3, 0C4, 2A12, 3A8, 3A9, 3B12, 3B14, 3B18, 3B2, 3C1, 3C3, 3C6, 4B3, 5C2, 5C3, 5C4, 5D1, 5D3, 7B1, 7B5
02_04_00	2A1, 3A4, 3A6, 3A7, 4C1
11_01_00	2A1, 2A13, 2A17, 3A17, 3A18, 3A2, 3A6, 3A8, 3A9, 3B12, 3B14, 3B18, 3B2, 3C1, 3C3, 3C6, 3C7, 4B2, 4C1, 5C2, 5C3, 5D1, 5D3, 5D5, 6C1, 6C2, 7B5, 7B6, 7C7
11_14_00	3A4
03_00_00	4A4
02_03_00	2A1, 3A1, 3A4, 3A6, 3A7, 4A3, 4C1
02_02_00	2A1, 3A1, 3A4, 3A6, 3A7, 4A3, 4C1
02_01_00	2A1, 3A1, 3A2, 3A4, 3A7, 4A3, 4C1
11_11_00	3A1, 3A4, 3B18, 3B3, 4E1, 6A1, 6A2, 7C7
01_03_00	0C2, 2A12, 3A8, 3B12, 3B13, 3B2, 3C3, 3C6, 5C2, 5C4
11_15_00	3A4
02_00_00	0A1, 2A1, 2A10, 3A1, 3A2, 3A4, 3A6, 3A7, 4A1, 4A2, 4A3, 4A5, 4C1
02_06_00	3A4, 4C1
02_05_00	3A4, 4C1
11_05_00	3C4
11_12_00	3A1, 3A4, 3B18, 3B3
11_04_00	2A13, 3B12, 3C3, 3C4, 3C7
11_10_00	3A1, 3A4, 3B18, 6A1, 6A2, 7C7, 7C8

01_00_00	2A10, 2A12, 2A9, 3A13, 3A15, 3A16, 3A17, 3A18, 3A19, 3A20, 3A21, 3A22, 3A23, 3A24, 3A8, 3A9, 3B11, 3B12, 3B13, 3B14, 3B18, 3B19, 3B2, 3B20, 3B21, 3B22, 3C1, 3C3, 3C4, 3C6, 3C7, 3C8, 3C9, 4A3, 4A4, 4A5, 4B2, 4B3, 4D1, 5C1, 5C2, 5C3, 5C6, 5C7, 5C8, 5C9, 5D1, 5D10, 5D2, 5D3, 5D5, 5D7, 5D8, 5D9, 6C1, 6C2, 6C5, 6C6, 6C7, 6C8, 7B1, 7B5, 7B6, 7C6
01_00_01	3A13, 3C7, 4D1
01_06_00	5C4
02_08_00	4C1
11_00_01	3A2, 3B14, 4B2, 4B3, 6C3, 6C4
11_00_00	2A1, 2A12, 2A13, 2A16, 2A17, 2A18, 2C10, 2C2, 3A1, 3A15, 3A16, 3A17, 3A18, 3A19, 3A2, 3A20, 3A21, 3A22, 3A23, 3A24, 3A6, 3A8, 3A9, 3B12, 3B13, 3B18, 3B19, 3B2, 3B20, 3B21, 3B22, 3B3, 3C1, 3C3, 3C4, 3C7, 3C8, 3C9, 4A1, 4A2, 4A3, 4A4, 4A5, 4B2, 4B3, 4C1, 4D1, 4E1, 5C1, 5C2, 5C3, 5C4, 5C6, 5C7, 5C8, 5C9, 5D1, 5D10, 5D3, 5D5, 5D7, 5D8, 5D9, 6A1, 6A2, 6C1, 6C2, 6C3, 6C4, 6C5, 6C6, 6C7, 6C8, 7B1, 7B5, 7B6, 7C6, 7C7, 7C8
11_03_00	2A13, 3B12, 3B2, 3C3, 3C4, 3C6, 3C7
02_07_00	4C1
02_00_00A	3A3, 3A5
01_04_00	3A8, 3B13, 5C4
01_00_00B	1A2, 2B1
01_02_01	0C2, 0C4
11_00_00A	2A15, 2B7, 2B8, 2B9, 2C7, 2C8, 3A7, 4E2
01_00_00A	1A1, 1A2, 2A3, 2A4, 2A6, 2A7, 2B1, 2B2, 2B3, 2B4, 2B5, 2C1, 2C10, 2C2, 2C3, 2C4, 2C5, 2C6, 2C7, 2C8, 2C9, 3A10, 3A14, 3B1, 3B11, 3B3, 3B4, 3B5, 3B6, 3C2, 3C5, 4A6, 5C5, 5D4, 5D6, 7B2