Predicting Change Propagation Impacts in Collaborative Business processes*

Walid Fdhila
University of Vienna
Faculty of Computer Science
Austria
walid.fdhila@univie.ac.at

Stefanie Rinderle-Ma
University of Vienna
Faculty of Computer Science
Austria
stefanie.rinderle-ma@univie.ac.at

ABSTRACT

During the life cycle of a Business-to-Business (B2B) collaboration, companies may need to redesign or change parts of their service orchestrations. A change request proposed by one partner will, in most cases, result in changes to other partner orchestration. An accurate prediction of the behavior of a change request and an analysis of its impacts on the collaboration allows to avoid significant costs related to unsuccessful propagation, e.g. negotiation fail. This paper focuses on predicting the likelihood of a change request propagation as well as its ripple effects on the overall collaboration. To estimate these values, the approach analyses the collaboration structure through a priori analysis. We will show how the prediction models can be specified and implemented within a proof-of-concept prototype. Discussion will be provided on visualization possibilities and model validation.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

Keywords

Prediction, change propagation, impact analysis, collaborative business processes

1. INTRODUCTION

Process flexibility has been identified as key concern in the Business Process Management (BPM) area [1]. While mature solutions to provide design and runtime time flexibility have been developed for single service orchestrations [15], flexibility and change in collaborative settings have only recently been paid attention to [10]. However, different application scenarios demand for enabling flexibility in collaborative orchestrations as well. Examples comprise virtual factories [17] and supply chains [16]. Figure 1 depicts the collaboration, i.e., choreography model of a trip booking process [3] that describes the interactions between four partners `Traveler`, `TravelAgency`, `Acquirer`, and `Airline` with the corresponding process activities as well as the control flow of the process. Figures 3 and 2 illustrate the views on the orchestration of partner `Acquirer`. Its private view (cf. Fig. 3) contains all peculiarities which are abstracted to the other partners by providing a public view (cf. Fig. 2) for confidentiality reasons.

As argued before, enabling partners to adapt their orchestrations when participating in a collaboration is of high need in many applications. The specific challenge when compared to single orchestrations is to *propagate* change effects from one partner to the other [10, 16]. Assume, for example, that the `TravelAgency` wants to send a questionnaire about customer satisfaction after booking the ticket to the `Traveler`. This would be accomplished by inserting associated activities into the private orchestration of the `TravelAgency`, e.g., `DevelopQuestionnaire` (not visible to other partners) as well as `SendQuestionnaire` (in the public process) and a respective *change request* to `Traveler` who should be able to receive the corresponding message and respond to it.

Change propagation in collaborative orchestrations might become quite complex [10]. Assume that in the example above, not the `TravelAgency` initiates collection of customer feedback, but the `Airline` through the `Acquirer` and the `TravelAgency`. In this case the initial change will cause *transitive* effects over several partners. As handling the effects of change propagation at the partners' sides can be neither enforced nor checked automatically as the information on the private partner processes is not available, change propagation might necessitate tenacious negotiations between partners [10]. A change propagation that fails in the end after many successful negotiations is costly and hampers the success of the collaboration. Hence, being able to predict the change request behavior of partners in order to assess *change impacts* on the choreography beforehand would be of great importance. Generally, change impact analysis enables developers and project leaders to ask "what if...?" questions, and to simulate alternative scenarios without having to implement them [18] and allows to judge the amount of work required to implement a change [4]. Particularly for collaboration of service orchestrations, predicting the results of negotiations that could be affected by the change might

*The work presented in this paper has been funded by the Austrian Science Fund (FWF):I743.

SAC'14 March 24-28, 2014, Gyeongju, Korea.

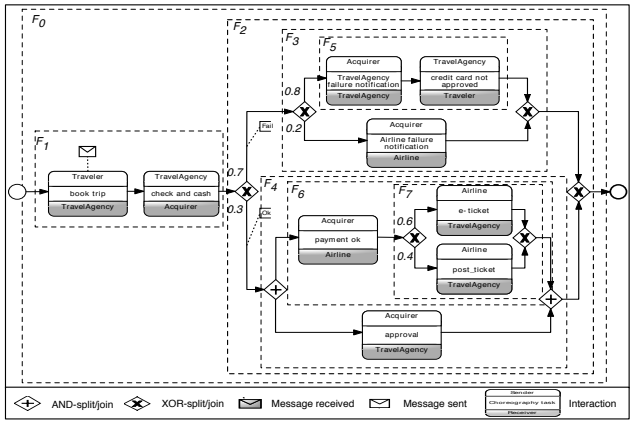

Figure 1: Choreography Model: Book Trip Operation [3]

avoid unsuccessful negotiations and the overall cost of a change request before initiating negotiation could be estimated. Further on, change impact analysis could support the redesign of orchestrations by minimizing change propagation through preventative measures. In [5], the authors realized that retrofitting design changes (from a late requirement or from a delayed problem realization) costs about five times more than an early design change and contracts may have to be renegotiated for them.

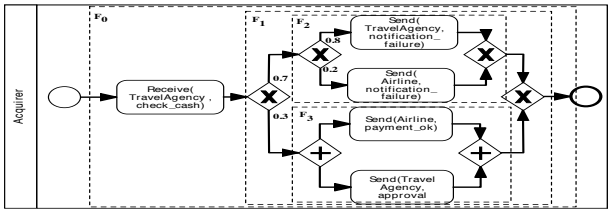

Figure 2: Public View

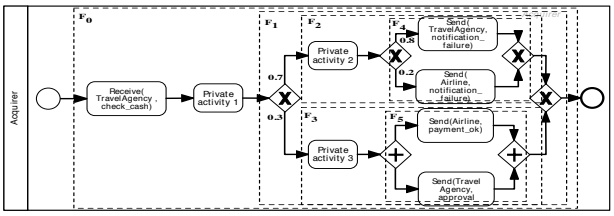

Figure 3: Private View

Change in collaborative settings has been mainly tackled by providing correctness criteria and algorithms for change propagation, e.g., in [10]. Change impact analysis has been studied for complex systems [4, 12, 13, 5, 18, 7, 6], but not in the context of collaborative orchestration settings so far. In this paper, we provide first a priori techniques for change impact analysis and prediction in collaborative business processes. More precisely, these techniques generate models that enable the prediction of ripple effects of a change request and estimates its impact on the partners within a choreography. A priori techniques do not necessitate any information about change requests and change propagation that have been applied in the past, but are purely based on choreography model information. As a result, partners participating in the choreography can be classified with respect to their probability to *absorb*, *carry on*, or *multiply* changes [6] providing a first risk assessment by means of a static

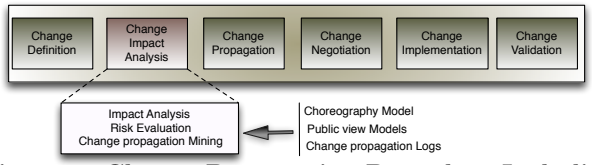

Figure 4: Change Propagation Procedure Including Change Impact Analysis

cost function. Assessing dynamic impacts, we evaluate the choreography structure equipped with information such as decision probabilities and number of running instances. This constitutes a major difference to change impact analysis in complex systems, taking into consideration information on orchestration executions.

Section 2 presents fundamentals. In Section 3, a priori techniques for change impact analysis are provided. Section 4 constitutes the evaluation of the approach. In Section 5, we compare our work to existing approaches for change prediction. Section 6 summarizes the contribution and outlines future work.

2. FUNDAMENTALS

Figure 4 describes the change propagation procedure in collaborative business processes (green boxes) as proposed in literature [8, 10, 16, 20]. After application of the change at a partner side (*Change Definition*), it is checked whether the change is just local or may affect other partners. If the change should be propagated, then all the partners affected directly or transitively by the change are identified, and the changes to be propagated to each of these partners are computed. Then, a negotiation process is initiated with all these partners in order to validate or cancel the change. If negotiations succeed, then soundness criteria (e.g. compatibility, consistency, etc.) are checked to see if the choreography would stay sound after the propagation of the changes. If so, then the changes are implemented and the process models as well as the choreography model are updated. Correctness criteria are checked after negotiation because some transitive effects could not be determined by the initiator, and also because the negotiation may lead to a modification of the initial change request itself. In this procedure, the change propagation and the negotiation phases could have significant costs. As emphasized, the aim of this work is then to anticipate this negotiations and predict the impact of changes on the other partners (purple box).

In this paper, we adopt the Refined Process Structure Tree (RPST) [19] to represent process and choreography models. RPST provides a structured representation, where the models are considered as trees whose leaves represent activities or interactions respectively and whose internal nodes represent either sequence (SEQ), parallel (PAR), choice (CHC), repeat loop (RPT), or while loop (WHILE) constructs. The elements inside a *repeat loop* are executed at least one time, while the elements inside a *while loop* are executed zero or more times. Structured models are simpler to analyze and easier to comprehend, and recent work has shown that most unstructured process models can be automatically translated into structured ones [14]. In the following, we distinguish between the choreography model $\mathcal{G}$ (c.f. Figure 1), the local model (also called public view) of a partner p denoted $\mathcal{L}_p$ (c.f. Figure 2), and its executable process model (called private view) denoted π_p (c.f. Figure 3).

DEFINITION 1. *[(Structured) Choreography Model] A choreography model $\mathcal{G}$ is a description of the coordinated interactions between all partners involved in the collaboration. It is defined as a* ***tree*** *with the following structure (type definition syntax of the ML language):*

$$
\begin{aligned}
ChoreographyModel &::= CNode \\
CNode &::= Interaction \mid ControlNode \mid Event \\
Interaction &::= I(Source, Destination, Message) \\
ControlNode &::= SEQ([CNode]) \mid CHC([P \times CNode]) \\
&\quad \mid PAR(\{CNode\}) \mid RPT(CNode \times P) \\
&\quad \mid WHILE(P \times CNode) \\
Event &::= Start \mid End
\end{aligned}
$$

where P is the range of real numbers from 0.0 to 1.0, denoting probabilities.

In the choreography example of Figure 1, the fragment $\mathcal{F}_3$ is represented by a choreography node ($CNode$) as follows: `CHC`(*0.8 ×* `SEQ`(`I`*(TA_failure_notification, acquirer, TravelAgency),* `I`*(credit_card_not_approved, TravelAgency, Traveler)*), *0.2 ×* `I`*(Airline_failure_notification, Acquirer, Airline)*)

DEFINITION 2. *[Local Public Model] A local public model $\mathcal{L}_p$ of partner p states the external behavior of p and is also denoted public view. It includes the interactions with other partners as well as the constraints between them from the viewpoint of this partner:*

$$
\begin{aligned}
LocalModel &::= LNode \\
LNode &::= Send(Message, Destination) \\
&\quad \mid Receive(Message, Sender) \\
&\quad \mid ControlNode \mid Event
\end{aligned}
$$

where $ControlNode$ and $Event$ have the same definitions as in Definition 1.

DEFINITION 3. *[Change Operation] A change operation is a tuple (δ,σ) where σ is either the local public model or the global choreography model to be changed, and δ: $\sigma \mapsto \sigma'$ is the change pattern that transforms σ into σ'. A change pattern allows to insert, delete, or replace a fragment in a given process model.*

Figure 5 represents a subset of change patterns as defined in [10, 16, 20]. As emphasized, a change operation (δ,σ) alters the source model σ to a new model σ'. In a collaboration context, this change could have impacts on the other collaborating partners. The problem is then to propagate this well-behaved change δ on σ to the collaborating partners σ_i, such that the updates on the different partners affected by the change lead to a consistent collaboration.

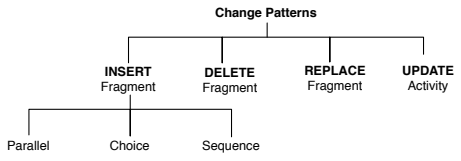

Figure 5: Change patterns

DEFINITION 4. *[Change Propagation] We define change propagation as a function $\varkappa$: (δ,σ) $\mapsto \{(\delta_i, \sigma_i)\}^{i=1..n}$ such as $\forall i, \delta_i \neq \delta$. The propagation function takes a change operation on a partner's orchestration and computes the exact effects on the different collaborating partners. For more details about change propagation in collaborative orchestrations, the reader may refer to [10, 16, 20, 9, 8].*

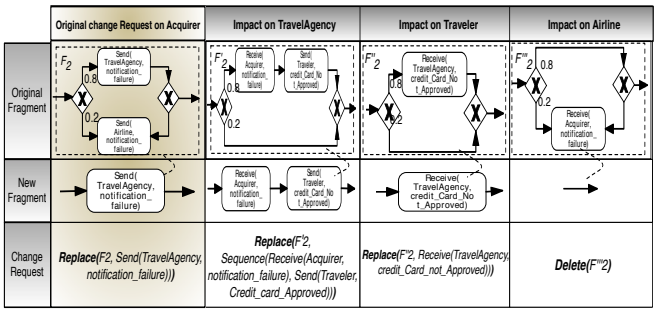

Figure 6: Book Trip Operation: Change Propagation Scenario

Figure 6 illustrates a simple example of change propagation, where the change is initiated by the `Acquirer` and propagated to the other partners. The initial change request consists in replacing $\mathcal{F}_2$ by a single activity (first column). The computation of the exact effects of the initial change request on the `Acquirer` leads to the propagation $\varkappa$ of the derived changes to the `TravelAgency`, the `Traveler` and the `Airline`. The third line of the table corresponds to the changes requests as defined previously. In this example, the impact on the `Traveler` was propagated transitively through the `TravelAgency` since there is no direct dependency between the `Traveler` and the `Acquirer` in the choreography model. This transitivity makes the propagation more complex and could not be calculated directly by the change initiator. Besides, the change requests negotiation as well as the transitivity effects could have significant costs. Thus the need for a prediction model to avoid costly unsuccessful propagations.

DEFINITION 5. *[α function] Let l_p be a local public model, $\mathcal{G}$ a choreography model and $\mathcal{F} \in l_p$ a single entry single exit fragment of l_p. Then, $\alpha_{\mathcal{G}}(\mathcal{F})$ is a function that returns the smallest fragment that contains all the corresponding elements of $\mathcal{F}$ in the model $\mathcal{G}$. We also consider a mapping function $f : l_p \rightarrow \mathcal{G}$ such as $\forall LNode \in l_p \backslash \{ControlNode\}$, $\exists I \in \mathcal{G} / f(LNode) = I$.*
$\alpha_\pi(\mathcal{F}) = Min_{size(\mathcal{F}')}\{\mathcal{F}' \in \mathcal{G} / \forall\ o \in \mathcal{F}, o \in \mathcal{F}'\}$

The smallest fragment that encapsulates the fragment $\mathcal{F}_2$ of the `Acquirer`, in the global choreography model $\mathcal{G}$ is $\alpha_{\mathcal{G}}(\mathcal{F}_2 \in \mathcal{L}_{acquirer}) = \mathcal{F}_3 \in \mathcal{G}$ (cf. Figure 1)

3. CHANGE IMPACT ANALYSIS

In this section, we present techniques for predicting change propagation behavior in collaborative business processes. The main idea is to create a propagation model which allows the anticipation of further change requests impact. This work is mainly inspired from existing research and studies on change impact analysis in software and complex systems, and analyzes the applicability and transferability of the proposed techniques to process choreographies. In the following, we present two complementary techniques for analyzing change propagation behavior: **(i)** a static impact analysis which focuses on the choreography structure and uses the choreography model and the public views in order to asses the probability of propagating changes to other partners as well as the respective costs. **(i)** a dynamic impact analysis which analyzes the effects of a change on the collaboration behavior at run-time. This theoretical study uses mainly the data

and the control dependencies between partner interactions as well as information about the running instances (e.g. the status of an instance).

3.1 Static Impacts

This prediction is based on an analysis of the choreography structure, the exchanged messages and the changed fragments. In particular it studies the likelihood of propagating a change to another partner as well as the impact of this change on the partner. This could be represented as a graph with the partners as vertices, and the probability of transition from one partner is the likelihood of propagation. Each vertex has a value which represents the impact of changing the corresponding partner. If we consider (δ_i, l_{p_i}) as a change request on the public model of p_i, then the corresponding estimated cost over the choreography is given as follows.

$$static\_cost_i = \sum_j c_j \times P((\delta_i, l_{p_i})/(\delta_j, l_{p_j})) \quad (1)$$

where $P((\delta_i, l_{p_i})/(\delta_j, l_{p_j}))$ represents the probability that a change on l_{p_i} would be propagated to l_{p_j}, and c_j the estimated cost for changing l_{p_j}. The propagation probability between two partners p_i and p_j could be estimated through their *dependency* in terms of interactions I (cf. Equation 2). $P((\delta_i, l_{p_i})/(\delta_j, l_{p_j}))$ is then given by calculating the transitive closure of the *dependency* matrix.

$$d_{ij} = \frac{\sum_{I \in \mathcal{G}} (\zeta_{ij}(I) + \zeta_{ji}(I))}{\sum_{I \in \mathcal{G}, k \neq i} (\zeta_{ik}(I) + \zeta_{ki}(I))}$$
$$with\ \zeta_{ij}(I) = \begin{cases} 1 & if source(I) = p_i\ and\ dest(I) = p_j \\ 0 & otherwise \end{cases} \quad (2)$$

To compute this, we start by identifying all the dependencies between the partners. It should be noted that, in a choreography, a partner p_i may have several interactions with a same partner p_j, and with various message importance $\varphi(m)$. φ is a function which assigns a value to an exchanged message based on its size, its sensitivity, etc. Equation 2 measures the dependency between two partners as a fraction between the number of times p_i sends or receives a message from p_j, and all interactions I involving p_i. This equation could also be weighted by the importance of the exchanged messages $\varphi(m)$.

In order to compute the cost c_j of propagating a change request to a partner p_j, we use two different attributes: (i) the centrality and (ii) the risk factor.

3.1.1 The centrality ψ

The centrality of a partner p is a function which evaluates the importance of p in the collaboration by measuring its capacity in propagating change requests. A partner with high centrality has a higher probability to propagate changes, in contrast, a partner with low centrality has more probability to absorb the changes. Different mathematical techniques introduced in graph theory or social networks could be used to measure the centrality of a given node [11, 13] such as: degree centrality, eigenvector centrality, betweenness centrality, page-rank or closeness centrality. For instance, Eigenvector centrality is based on the idea that a relationship to a more interconnected node contributes to the own centrality to a greater extent than a relationship to a less well interconnected node. In graph theory, this means that a node can acquire high centrality either by being connected to a lot of other nodes, or by being connected to others that themselves are highly central. In the context of business choreographies, this could be used in order to determine the capability of a partner in propagating changes. The higher he is central, the most he propagates changes and the overall cost for the initial change increases.

	A	TA	T	Ar
$Acquirer(A)$	0	0.6	0	0.4
$TravelAgency(TA)$	0.44	0	0.28	0.28
$Traveler(T)$	0	1	0	0
$Airline(Ar)$	0.5	0.5	0	0

Table 1: Dependency Matrix

Partner	Centrality
$Acquirer(A)$	0.27
$TravelAgency(TA)$	0.29
$Traveler(T)$	0.07
$Airline(Ar)$	0.19

Table 2: Centrality: Eigenvector Values

Table 1 represents the dependency matrix between the partners of the book trip operation example, while Table 2 represents the centrality matrix based on Eigenvector values. The latter uses the first table and, through an iterative algorithm, computes the centrality of each of the partners (for more details about Eigenvector algorithm the reader may refer to [13]). Here, we remark that the `TravelAgency` is very central and has high propagation factor in contrast to the `Traveler` which has low centrality.

3.1.2 The risk factor $\mathcal{R}$

In a choreography, several compliance, security or privacy rules could be shared by different partners. These rules are presented as constraints on activities of different partners. Therefore, a change on one partner process may invalidate or violate some of these rules and lead to a renegotiation of the agreements. In this sense, a partner which is implied in several rules is more subject to violate some of them if his process has to be changed. Then, the risk factor $\mathcal{R}$ is computed according to the number and the importance of the constraints a partner is involved in, and helps to evaluate the risk of violating these constraints. If we consider β_{ij} as a function that returns the number of constraints shared between two partners p_i and p_j, then the risk of violating p_i's constraints is calculated as a fraction between the total number of constraints involving p_i, and the number of interactions it has. This ratio could be superior to 1, and the higher it is, the more the probabilty of violating constraints on the corresponding partner increases.

$$\mathcal{R}_i = \frac{\sum_{j \neq i} \beta_{ij}}{\sum_{I \in \mathcal{G}, k \neq i} (\zeta_{ik}(I) + \zeta_{ki}(I))} \quad (3)$$

Business designers are able to influence the relative importance given to the risk factor $\mathcal{R}$ versus the centrality ψ of the partner affected by the change, by setting two weights. Then, the impact of propagating change to a given partner p_j is a weighted product of the costs related to the defined attributes:

$$c_j = w_1 * \mathcal{R}_j + w_2 * \psi_j \quad with \sum w_i = 1 \quad (4)$$

3.2 Dynamic Impacts

In general, the choreography model is not executable, but serves as a contract to monitor the execution of the collaborating processes; i.e. private processes. It also gives a global view on the interactions between all the partners. In this section, we estimate the impact of a partner change request on the running instances over the choreography. During

runtime, each partner executes his process in coordination with the other partners and following the agreements defined by the choreography (in addition to the compliance rules). Therefore, several instances run simultaneously and are executed by the corresponding partners according to the case scenarios. Let an instance represent the execution scenario of the whole choreography and not from a viewpoint of a single partner, i.e., a single instance is executed by several collaborating partners and its execution status is defined by its status in the choreography model. In the following, we consider only structural effects of changes to measure the impact of a change request on running instances, but the approach could be extended with an analysis of semantic effects. Even though the choreography model is not executable, we use the term execution of an interaction in the choreography model to refer to the execution of the corresponding *send* and *receive* activities on the actual executable orchestrations.

Consider N as the average number of running instances and $\mathcal{F}$ as the fragment to be changed in the partner's local public model $\mathcal{L}_p$. $\alpha_{\mathcal{G}}(\mathcal{F})$ represents the smallest fragment that encapsulates all the interactions in $\mathcal{F}$, in the global choreography model $\mathcal{G}$ (*cf. Definition 5*). The instances that could be affected by the change are those which according to their status, are executing or already executed $\alpha_{\mathcal{G}}(\mathcal{F})$. Then, the dynamic impact of a change request is as follows.

$$\begin{aligned} Dynamic\_Cost =& N \times [P(\mathcal{S} \in \alpha_{\mathcal{G}}(\mathcal{F})) + P(\mathcal{S} > \alpha_{\mathcal{G}}(\mathcal{F}) \backslash \alpha_{\mathcal{G}}(\mathcal{F}))] \\ & \times AdaptationCost \end{aligned} \tag{5}$$

- where $\mathcal{S}$ is a discrete random variable and corresponds to the status of an instance in the choreography model. The possible values of the random variable $\mathcal{S}$ is equal to the set of all interaction nodes in $\mathcal{G}$; i.e. $\{I / I \in \mathcal{G}\}$.
- $P(\mathcal{S} \in \alpha_{\mathcal{G}}(\mathcal{F}))$ is a probability distribution function for discrete random variables. It defines a distribution of the instances status over the choreography model. In particular, it considers the instances that are still executing interactions $I \in \alpha_{\mathcal{G}}(\mathcal{F})$.
- $P(\mathcal{S} > \alpha_{\mathcal{G}}(\mathcal{F}) \backslash \alpha_{\mathcal{G}}(\mathcal{F}))$ is a probability distribution function of the instances which already executed $\alpha_{\mathcal{G}}(\mathcal{F})$ and have not reached the *end* event yet. It is possible that the status of a given instance in the choreography model comes after $\alpha_{\mathcal{G}}(\mathcal{F})$, without having executed $\alpha_{\mathcal{G}}(\mathcal{F})$ (e.g. the case of a XOR). In this case, the instance does not need to be adapted to the new model since it is not concerned by the change. Only instances that executed or still executing $\alpha_{\mathcal{G}}(\mathcal{F})$ need to be adapted.
- *AdaptationCost* is the cost for migrating, or restarting running instances affected by the change [15]. The adaptation of an instance depends mainly of the executed interactions (i.e. visited path) from the root node until $\mathcal{S}$.

3.2.1 Computing probabilities $P(\mathcal{S} \in \alpha_{\mathcal{G}}(\mathcal{F}))$

The nodes in a choreography model do not have the same probability of execution during runtime. Then, the nodes with high probability of execution have more chance to be executed by the running instances. If we consider the paths between the *start* and *end* events, then a path with a high aggregated execution probability over the nodes it contains (i.e. the average of execution probabilities of its nodes) has more chance to be visited by the running instances, than the ones with low probabilities. In this sense, the distribution of the running instances is not equitable over the different execution paths, but depends of the execution probabilities of the nodes in each of these paths. For example, let's assume two different paths ρ_1 and ρ_2 linking *start* to the *end* event, and have as aggregated probabilities 0.1 and 0.9 respectively. Then, if at time t there are N instances on the *start* event, then the probability at time $t' > t$ that these instances are distributed over the nodes of ρ_2 is equal to 0.9.

It should be noted that the nodes on a same path do not have the same probability of holding an instance status at a time t. Indeed, two different paths ρ_1 and ρ_2 could share a subset of nodes $\omega = \{I \in \rho_1 \cap \rho_2\}$. For N instances, the probability of having instances with status in nodes of ω is higher than the probability over the other nodes. This is due to the fact that the nodes of ω could be reached through different paths. Through these two observations we can conclude that the distribution of the instances through the choreography model depends mainly on the execution probability of each interaction. If we consider $P_{exec}(\mathcal{S} = I)$ as the probability that the discrete random variable $\mathcal{S}$ has the interaction I as value (i.e. the status of the instance is I), then:

$$P(\mathcal{S} \in \alpha_{\mathcal{G}}(\mathcal{F})) = \sum_{I \in \alpha_{\mathcal{G}}(\mathcal{F})} P(\mathcal{S} = I) \tag{6}$$

where $P(\mathcal{S} = I)$ is a probability function of execution of I, i.e. $P_{exec}(I)$. The execution probability of I in the choreography tree model (RPST) (c.f. Def. 1) is given by Alg. 1. We traverse the tree model from node I in the root (*start* event), and for each traversed conditional control flow arc (XOR gateway), the execution probability is multiplied by the probability attached to the latter. For each traversed loop, the execution probability is multiplied by $\frac{1}{1-(1-P)P_l}$ if it is a repeat loop and by $\frac{P_l}{1-(1-P)P_l}$ if it is a while loop.

Proof: If P is the probability of executing I inside the loop, then $(1-P)$ corresponds to the probability that I would not be executed. Assume P_l is the probability of staying in a **repeat** loop, then in order to execute I **one time**, we have two possibilities: (i) executing directly I with the probability P, or (ii) looping a number of times on the paths that do not pass by I inside the loop and executing I at the next iteration. This is formulated as follows. $P_{exec}(I) = P + (1-P)P_l P + (1-P)^2 P_l^2 P + ... + (1-P)^n P_l^n P + ... = P \sum_i ((1-P)P_l)^i = P \times lim_{n\to\infty} \frac{1-((1-P)P_l)n}{1-(1-P)P_l} = \frac{P}{1-(1-P)P_l}$ (geometric series).

3.2.2 Computing probabilities $P(\mathcal{S} > \alpha_{\mathcal{G}}(\mathcal{F}) \backslash \alpha_{\mathcal{G}}(\mathcal{F}))$

This distribution function concerns only the instances which have already executed $\alpha_{\mathcal{G}}(\mathcal{F})$ and have not reached the *end* event. It could be obtained using the formula of *Bayes*, by multiplying $P(\mathcal{S} > \alpha_{\mathcal{G}}(\mathcal{F}))$ by the probability of reaching the status S from the exit node of $\alpha_{\mathcal{G}}(\mathcal{F})$. The procedure is similar to the Algorithm 1.

3.2.3 Computing *AdaptationCost*

In general, the adaptation cost is different from one instance to one another according to their run-time status which could be known through real time monitoring [2]. However, since this is a *priori* assessment of change impact

Algorithm 1: Execution Probability of an Interaction

```
1  Input: -G /*Choreography tree Model*/, I /*an Interaction in G*/
2  begin
3      current ← the parent node of I in the Rpst model of G;
       P ← 1;
4      while current ≠ root do
5          if current = cb ∈ conditionBranches then
6              P ← P × cb
7          else if current = l ∈ loops then
8              P ← P / (1−(1−P)P_l)  // if l is a repeat loop;
               P ← P_l P / (1−(1−P)P_l)  // if l is a while loop;
9          end
10         current ← carrent.parent
11     end
12 end
```

on running instances, we consider an average adaptation cost according to the discrete distribution function presented previously. The adaptation cost depends mainly on the number of activities to be migrated weighted by their execution number. The execution number of an interaction I in a loop free model is equal to its execution probability. In a presence of loops this number depends on the type of the loop; i.e. a repeat loop or a while loop. For instance, let's consider l as a *loop* in the process model, with P_l as the probability of staying in the loop, and $\mathcal{F}_l$ as the fragment containing all interactions encapsulated by the loop.

- Repeat loop: the repeat loop induces at least one execution of $\mathcal{F}_l$. For each iteration of a repeat loop corresponds exactly one execution of $\mathcal{F}_l$. Then the number of executions of $\mathcal{F}_l$ is given as follows.

$$Nb_{exec}(\mathcal{F}_l) = \frac{1}{1-p_l} \tag{7}$$

Proof: Since l is a repeat loop, then $\mathcal{F}_l$ is at least executed one time. The probability of executing $\mathcal{F}_l$ a second time is given by the probability of staying in the loop after the first execution P_l. At the nth iteration, the probability of executing $\mathcal{F}_l$ is given by $\prod_{i=1..n} P_l = (P_l)^n$. Then, the total number of executions of $\mathcal{F}_l$ is considered as the sum of the execution probability of $\mathcal{F}_l$ for all the iterations i.e. $1 + P_l + P_l^2 + ... + P_l^n + ... = \sum_{i=0}^{\infty}(P_l)^i$. The last equation represents an infinite geometric series with as a result $lim_{n\to\infty}\frac{1-(P_l)^n}{1-P_l} = \frac{1}{1-P_l}$.

- While loop: the while loop induces zero or more executions of $\mathcal{F}_l$ since the loop condition is examined before the execution of the loop's elements. Then the number of executions of $\mathcal{F}_l$ is given as follows.

$$Nb_{exec}(\mathcal{F}_l) = \frac{p_l}{1-p_l} \tag{8}$$

Proof: The difference with the repeat loop is that the first execution of $\mathcal{F}_l$ is equal to P_l instead of 1. Then the total number of executions of $\mathcal{F}_l$ is: $P_l + P_l^2 + ... + P_l^n + ... = (\sum_{i=0}^{\infty}(P_l)^i) - 1$ $= lim_{n\to\infty}\frac{1-(P_l)^n}{1-P_l} - 1 = \frac{1}{1-P_l} - 1 = \frac{P_l}{1-P_l}$

The equations 7 and 8 represent estimations of the number of times the whole bloc inside the loop (i.e. $\mathcal{F}_l$) would be executed. However, $\mathcal{F}_l$ is a non empty set of interactions, fragments, and possibly loops. Then, the execution number of a single interaction I in the loop $l$is calculated as the its execution number inside $\mathcal{F}_l$ multiplied by the execution number of $\mathcal{F}_l$.

- If $\mathcal{F}_l$ is loop free then the probability of execution of an interaction inside $\mathcal{F}_l$ is given by multiplying the probabilities of the conditional branches *cb* that appear in the path from the entry of the loop to the interaction I. $P_{exec}(I \in loop) = \prod_{cb\in CondBranches} P(cb)$.
- if $\mathcal{F}_l$ is contains nested loops such that I is encapsulated by a subset of them, then we employ a recursive method starting by the most inner loop.

Assume that *co* represents the average adaptation cost of one interaction. The adaptation cost of an instance depend on the status of the latter in the choreography model. If the status $\mathcal{S}$ is inside $\alpha_{\mathcal{G}}(\mathcal{F})$, then the cost is quantified by the execution number of all nodes on the critical path in $\alpha_{\mathcal{G}}(\mathcal{F})$, i.e. the path with highest aggregated number of executions (according to the nodes it contains). If the status $\mathcal{S}$ is outside $\alpha_{\mathcal{G}}(\mathcal{F})$ but already executed $\alpha_{\mathcal{G}}(\mathcal{F})$ before, then *co* is quantified by the critical path from the entry node of $\alpha_{\mathcal{G}}(\mathcal{F})$ until $\mathcal{S}$. If $\mathcal{S}$ did not execute $\alpha_{\mathcal{G}}(\mathcal{F})$ then the adaptation cost is equal to zero. The dynamic and static impact analyses give an overview on the possible ripple effects of a given change in a collaboration and its impacts on the running instances.

4. PROOF-OF-CONCEPT AND USAGE

We assume that the probabilities of the choice and repeat patterns as well as the average number of running instances are known. They could be computed based on the statistics gathered from the monitoring of the collaboration's normal execution. The approach can be extended with new attributes that enhance the risk assessment of changing a part of the choreography.

4.1 Implementation

We have implemented the proposed techniques for impact analysis and integrated them with our prototype for change propagation in the context of the C^{3}Pro project[1]. The tool takes as inputs a set of collaboration models; i.e. interconnected public models l_p, and the set of corresponding choreography models described in BPMN, and generates the prediction models as described throughout the paper. The experiments were conducted using the models defined in [3]. We also used Signavio[2] to edit the models in BPMN and annotate them with branching probabilities and costs. The models were exported to our tool as xml files and transformed into RPST representations (as defined in Section 2) using the JBPT library[3]. Our experimental setup uses an Intel processor Core i7, 2.2 GHz, a 4 GB memory and Eclipse Juno environment. The generated results are directed graphs with the partners as vertices. The edges are weighted with the probability of propagation. The vertices are annotated with their centrality degree and the average costs of changing the correspondent partner model. For the dynamic impacts we considered that the probability that a status of an instance is executing a given interaction as equal to the execution probability of I. We assumed that the adaptation cost of an affected instance is equal to the sum of the execution number of each activity affected by the change multiplied by its cost (the cost is edited manually with Signavio). For each collaboration model, we gener-

[1]http://www.wst.univie.ac.at/communities/c3pro
[2]http://academic.signavio.com
[3]http://code.google.com/p/jbpt/

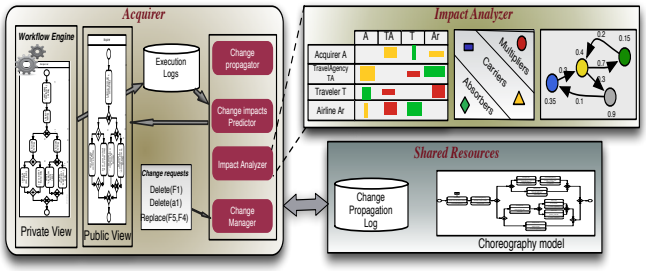

Figure 7: Change Propagation Architecture

ated automatically a set of random change scenarios on each partner, and for each scenario we used the change propagation prototype to compute the exact propagation. Then, we gathered all the propagations logs and computed statistics related to: (i) the number of times a partner is affected by the propagation through all change scenarios (i.e. the centrality) , and (ii) the number of times a partner is affected by the propagation following a change initiated by an another partner (i.e. the dependency matrix). Then, we compared the results with the prediction algorithms proposed in this paper. The results proved a very similar behavior in the propagation to the exact algorithms. Table 3 presents the results obtained by the propagation of 198 change scenarios on the book trip operation example presented in the paper. Compared with Table 1 and 2, the propagation probability between pair of partners is to a certain extent similar as well as the centrality measures. However, the centrality of the `airline` is lower than the `TravelAgency` in the prediction (0.19) but higher in the exact algorithms (0.33). This is mainly due to the change operations of type Insert which can add new interactions with different partners and consequently augment their centrality.

	A	TA	T	Ar	Centrality
$Acquirer(A)$	0	0.5	0	0.5	0,23
$TravelAgency(TA)$	0.37	0	0.25	0.37	0,29
$Traveler(T)$	0	1	0	0	0,12
$Airline(Ar)$	0.33	0.66	0	0	0,33

Table 3: Change propagation Results

4.2 Usage:

The results of the impact analysis can be used to, for example, categorize the business partners according to their behavior for change propagation, and assessing the magnitude of a change request based on several criteria. For this purpose, we transfer existing classifications of nodes in complex product structures with respect to their change behavior [7] to the partners participating in a choreography.

- *Constants:* not affected by the changes. They do not appear in the propagation model.
- *Absorbers:* they propagate fewer changes than they are impacted by. In the propagation model, the probability of propagating is less than the probability of receiving change requests: $\sum_{ij} P(\mathcal{L}_i/\mathcal{L}_j) - \sum_{ij} P(\mathcal{L}_j/\mathcal{L}_i) < 0$.
- *Carriers:* they propagate as many changes they are impacted by: $\sum_{ij} P(\mathcal{L}_i/\mathcal{L}_j) - \sum_{ij} P(\mathcal{L}_j/\mathcal{L}_i) = 0$.
- *Multipliers:* they propagate more changes than they are impacted by: $\sum_{ij} P(\mathcal{L}_i/\mathcal{L}_j) - \sum_{ij} P(\mathcal{L}_j/\mathcal{L}_i) > 0$.

As each change and its propagation causes effort and complexity, particularly *multipliers* can be critical regarding the effect of change propagations. Hence, identifying partners as multipliers can help, for example, to avoid propagating changes to them by restructuring the initial change request and privilege propagation to carriers or absorbers. Besides, the prediction models allow to measure the magnitude of a change request by following the possible propagation paths scenarios and considering the predicted impacts on the different partners. These results could be visualized and analyzed using visualization techniques [5, 12].

4.3 Architectural Considerations

Figure 7 sketches an architecture for change management in collaborative process scenarios, i.e., we provide an integrated view on usual change propagation techniques and the results of this paper. Each partner maintains two models: the private and public views, and share the global choreography model. The private view represents its executable process, should be consistent with its public view, and contains additional private activities. Each partner maintains four components; i.e. change manager, change propagator, change impacts predictor, and impact analyzer. The first component is responsible for defining, implementing, validating (e.g. correctness) and managing changes from the point of view of a single partner. The second one is responsible for calculating the partners affected by the change as well as the changes to propagate to each of them, and for the negotiation part. The change predictor predicts the behavior of a change request and assesses its impact on the choreography. It also estimates if a change request would be achieved or not, and prevents several costs related to unsuccessful negotiation. The change predictor component can be extended by further change impact analysis techniques, for example, based on change propagation logs. The prediction and impact analysis components are independent from the change propagation methods and depend only on the choreography structure and the different constraints of the collaboration (e.g. compliance rules).

5. RELATED WORK

Change impact analysis is required for constantly evolving systems to support the comprehension, implementation, and evaluation of changes [18] and has been widely studied in different domains, in particular, in large complex systems and software engineering [4, 12, 13, 5, 18, 7, 6]. In [18], a literature review of 150 studies about impact analysis in software systems was investigated and a classification based on the evaluation of the taxonomy and its criteria was proposed. In [12], an analysis for change propagation in complex sensor systems using a data set of 41,500 change requests is presented. The authors use design documentation in order to create a change design structure matrix DSM to represent the change propagation structure, and introduce three relationship types between the changes requests (Motifs) in order to analyze the change networks. They also use three indices to quantify each area in the system in terms of its propensity for accepting, reflecting or propagating changes. In [5, 7], the authors present an analysis of change behavior based on a case study in GKN Westland Helicopters. In particular, they discuss prediction and management of changes to an existing product resulting from faults or new requirements. The proposed approach is based on math-

ematical models to predict the likelihood and impact of a change. These approaches predict change impacts either in large complex systems or in software systems. This work is inspired by them, but analyzes the applicability and transferability of these techniques to process choreographies. The problem is different due to the structure of the choreography and the additional constraints it has in terms of compliance, privacy, or running instances.

In the context of web service choreographies, only few papers tackled the impact analysis of changes propagation. In [13], the authors present a prototype for dynamic adaptation in choreographies. Adaptations are mainly reconfigurations of service endpoints. The aim of the prototype is to support dynamic reconfiguration of collaborative BPEL processes, and to assess the impact of changing services. Their work is based on a use case defined in BPEL rather than a general approach for business process choreographies and they do not assess the risk of the propagation. In [21], the authors present a dependency and entropy based impact analysis model for service oriented systems evolution. This approach combines dependency analysis to measure the importance of a given service in the collaboration, and information entropy to allow quantification measures of the system. Both approaches consider only static impacts of changes and do not assess the dynamic effects on running instances. Moreover, they do not consider previous change propagation experience to enhance the prediction models. The model adopted in our work allows more precise analysis since it employs transition probabilities in control patterns and quantifies the loops effects, instead of using simple dependencies between the services.

6. CONCLUSION

Change propagation in collaborative process scenarios might lead to costly negotiations and increased complexity. Hence, estimating the effects of necessary change propagations on partners can be of crucial help. This paper focused on analyzing the choreography structure and the dynamic aspects of business process collaborations, to outline a change propagation model through a priori prediction technique. Based on the change propagation model, partners can be categorized along their behavior in case of change propagation. In particular, identifying partners that potentially multiply the propagation requests might avoid propagating changes to them by restructuring the initial change request and privilege propagation to carriers or absorbers. The presented technique has been implemented prototypically and its embedding in a change propagation framework has been discussed. In future work, we will elaborate a posteriori techniques exploiting the information on previous change propagations and we will investigate visualization techniques for results of change impact analysis.

7. REFERENCES

[1] W. M. P. v. d. Aalst. A decade of business process management conferences: Personal reflections on a developing discipline. In *BPM*, LNCS, pages 1–16. 2012.

[2] A. Baouab, W. Fdhila, O. Perrin, and C. Godart. Towards decentralized monitoring of supply chains. In *ICWS*, pages 600–607, 2012.

[3] F. M. Besson, P. M. Leal, and F. Kon. Towards verification and validation of choreographies. technical research report, University of São Paulo, 2011.

[4] S. A. Bohner and R. S. Arnold. Software change impact analysis. IEEE Computer Society, 1996.

[5] P. J. Clarkson, C. Simons, and C. Eckert. Predicting Change Propagation in Complex Design. *Journal of Mechanical Design*, 126(5):788–797, 2004.

[6] C. Eckert, W. Zanker, and P. J. Clarkson. Aspects of a better understanding of changes. In *ICED*, volume 1, 2001.

[7] C. M. Eckert, R. Keller, C. Earl, and P. J. Clarkson. Supporting change processes in design: Complexity, prediction and reliability. *Reliability Engineering and System Safety*, 91(12):1521–1534, 2006.

[8] W. Fdhila, A. Baouab, K. Dahman, C. Godart, O. Perrin, and F. Charoy. Change propagation in decentralized composite web services. In *CollaborateCom*, pages 508–511, 2011.

[9] W. Fdhila, S. Rinderle-Ma, A. Baouab, O. Perrin, and C. Godart. On evolving partitioned web service orchestrations. In *SOCA*, pages 1–6. IEEE, 2012.

[10] W. Fdhila, S. Rinderle-Ma, and M. Reichert. Change propagation in collaborative processes scenarios. In *CollaborateCom*, Pittsburgh, USA, October 2012.

[11] L. C. Freeman. Centrality in social networks conceptual clarification. *Social Networks*, pages 215 – 239, 1978.

[12] M. Giffin, O. de Weck, G. Bounova, R. Keller, C. Eckert, and P. J. Clarkson. Change propagation analysis in complex technical systems. *Journal of Mechanical Design*, 131(8), 2009.

[13] G. A. Oliva, G. de Maio Nogueira, L. F. Leite, and M. A. Gerosa. Choreography Dynamic Adaptation Prototype. Technical report, Universidade de São Paulo, 2012.

[14] A. Polyvyanyy, L. Garcia-Banuelos, and M. Dumas. Structuring acyclic process models. *Information Systems*, 37(6):518–538, 2012.

[15] M. Reichert and B. Weber. *Enabling Flexibility in Process-Aware Information Systems - Challenges, Methods, Technologies*. Springer, 2012.

[16] S. Rinderle, A. Wombacher, and M. Reichert. Evolution of process choreographies in DYCHOR. In *CoopIS*, LNCS, pages 273–290, 2006.

[17] S. Schulte, D. Schuller, R. Steinmetz, and S. Abels. Plug-and-play virtual factories. *IEEE Internet Computing*, 16(5):78–82, 2012.

[18] L. Steffen. A review of software change impact analysis. Technical report, Universitätsbibliothek Ilmenau, 2011.

[19] J. Vanhatalo, H. Vôlzer, and J. Koehler. The refined process structure tree. In *Business Process Management*, volume 5240, pages 100–115, 2008.

[20] M. Wang and L. Cui. An impact analysis model for distributed web service proces. In *Computer Supported Cooperative Work in Design (CSCWD)*, 2010.

[21] S. Wang and M. A. M. Capretz. Dependency and entropy based impact analysis for service-oriented system evolution. In *Web Intelligence*, pages 412–417, 2011.