Open access • Proceedings Article • DOI:10.1109/ICWS.2015.80

Paving the Way towards Semi-automatic Design-Time Business Process Model Obfuscation — Source link

Elio Goettelmann, Amina Ahmed-Nacer, Samir Youcef, Claude Godart

Institutions: French Institute for Research in Computer Science and Automation

Published on: 27 Jun 2015 - International Conference on Web Services

Topics: Business process modeling, Artifact-centric business process model, Business process, Model transformation and Context (language use)

Paving the Way towards Semi-automatic Design-Time Business Process Model Obfuscation

Elio Goettelmann, Ahmed-Nacer Amina, Samir Youcef, Claude Godart

Paving the way towards Semi-automatic Design-time Business Process Model Obfuscation

Elio Goettelmann[1,2], Amina Ahmed-Nacer[1,3], Samir Youcef[1] and Claude Godart[1]
[1]LORIA - INRIA Grand Est
Université de Lorraine, Nancy
France
[2]CRP Henri Tudor
L-1855 Luxembourg-Kirchberg
Luxembourg
[3]LIMED
Université Abderhaman Mira, Bejaia
Algeria
elio.goettelmann@tudor.lu, {*amina.ahmed-nacer, samir.youcef, claude.godart*}*@loria.fr*

***Abstract*—Business process (BP) stakeholders want to enjoy the benefits of the cloud, but they are also reluctant to expose their BP models which express the know-how of their companies. To prevent such a know-how exposure, this paper proposes a design-time approach for transforming a BP model into BP fragments so that these BP fragments externalized in a multi-cloud context do not allow a cloud resource provider to understand a critical fragment of the company. While existing contributions on this topic remain at the level of principles, we propose an algorithm supporting automatically such a BP model transformation.**

***Index Terms*—Business Process; Security Risk Management; Cloud; Privacy; Obfuscation**

I. Introduction

Cloud computing avoids upfront infrastructure costs, and helps organizations to focus on their core business activities, instead of their system infrastructure.

This concerns also business process (BP) execution. Companies have a long tradition of cross-organizational processes, especially in SaaS architectures. But in general such settings are well established between well-known cooperating business actors: the idea here is to go one step further with the externalization of BP fragments in the cloud with less established and as a consequence more risky cooperation links.

In fact, using off-premise and shared cloud infrastructures exposes the information systems of companies to new kind of security risks. And what is yet a problem in general is probably more exacerbated in the context of business processes which express the know-how of companies: they are ready to outsource their business processes to the cloud, but they want to preserve their know-how.

One way for companies to prevent risks is to transform their process models at premises so that one externalized BP fragment do not allow understanding a critical part of a BP model. This is connected to the idea of program obfuscation that makes code harder to understand or read, generally for privacy or security purposes. And in the same way that an obfuscator tool is sometimes used to convert a straight-forward program into one that works the same way but is much harder to understand [1], our objective is to develop means supporting BP models obfuscation.

In [2], we have elaborated a methodology for transforming and obfuscating a BP model before a trusted deployment in the cloud. But this obfuscation process is yet at the level of recommendations and the work has to be done mainly by designers' hands. The objective of this paper is to go one step further and to describe an approach for partly automating this obfuscation process.

The rest of this paper is organized as follows. Section II establishes the motivation and the context of this work. Then, section III characterizes the notion of a critical BP fragment, on which a particular effort must be paid for obfuscation. The following section explains how obfuscation is put in practice in our architecture. Section V discusses the state of the art and finally section VI concludes and introduces some future work.

II. Motivations and context

This section gives the motivation and the context of our work. It starts with an example illustrating some practical needs for process model obfuscation. Then it presents a synthesis of process obfuscation means and localizes the place where this paper contributes. Finally we situate the process obfuscator tool in a global architecture and explain how the obfuscator integrates with other components.

A. Motivating example

Fig. 1 depicts a loan process in a bank which objective is to accept or reject a loan request. Depending on the customer history and other parameters (loan amount, ...), the loan is treated in different ways. In general, the risk of the loan is evaluated, but the loan request can be either directly accepted or rejected. At any point in the process the hierarchy can directly intervene. The final decision is taken depending on the loan request treatment and the hierarchy validation.

The bank is ready to use cloud resources. However, it needs to be in confidence with its cloud providers, and especially to be sure that its strategy for directly accepting or rejecting a loan will not be disclosed. In the same way, it does not want to disclose how the hierarchy intervenes in the process and how the final decision is taken. One way for reaching this objective is to anticipate problems before they occur and the bank is ready to make some preliminary work in this direction. For mitigating risks, the bank can transform its BP model using the principles and the methodology introduced in [2] and overviewed in section II-B, and which consequence is to obfuscate BP models.

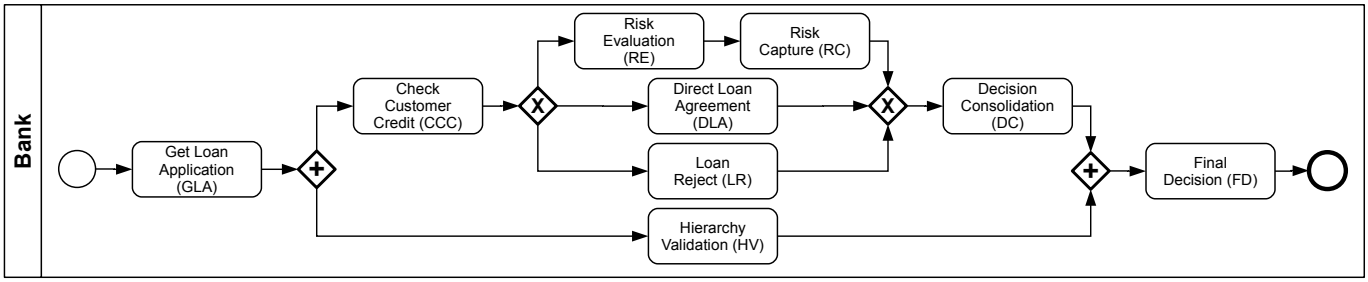

Fig. 1. A Loan Process

B. Process obfuscation means

The objective of the obfuscation activity is to preserve the know-how that BP models formalize. This encompasses different means [2] that this section overviews:

1) Retain sensitive data and logic at premises: Of course, one way to preserve the know-how contained in the process is not to disseminate it and to retain sensitive parts at premises. For example, in the *Loan* process example, designers can decide to maintain at premises *Check Customer Credit* and *Final Decision* tasks that contain important know-how.

2) Split BP logic into several BP fragment logics: Another intuitive mean is to split the BP logic in several fragments and to distribute them to different cloud providers so-that a cloud provider has only a partial view of the whole logic. In the *Loan* example, in the case designers do not decide to maintain *Check Customer Credit* and *Final Decision* on promises, these should probably be managed by two different cloud providers.

3) Add non-functional logic: Adding useless code in a program is a mean used by programmers for obfuscating it. In the same objective, useless BP fragments can be added. In this direction, we understood that some service logic added for non-functional purpose (security, replication for verification, ...) has also the property to increase the complexity of BP logic understanding.

4) Obfuscate data: Data obfuscation is also largely used to obfuscate programs. Cryptography is used to obfuscate data, but to execute, a task needs non encrypted data. Another mean which can be used with readable data is anonymization. It allows tasks working with neutralized data, and in our context to hide (partly) the link between data and logic. In the *Loan* process example, anonymizing customer information account when possible is a way to render the strategy of the bank for loan management more difficult to discover.

5) Separate logic and data: The idea there is to store the logic and its related data in different places so that a cloud cannot mine some links between data and logic by analysing logic and data storage.

6) Split cases between clouds: The objective is to split the process cases (instances) between the different clouds so that none of the clouds has enough data to mine the process logic (the number of different cases necessary to mine a BP model is easily calculable [3]).

In this work we focus on BPs logic splitting for program obfuscation. However, while deciding how to split a process is feasible for an informed designer having the semantics of the process in mind, it appears to be very difficult to automate this activity as it is concerned with process semantic. It is our objective in this paper to contribute to such automation.

C. Architecture

The architecture described in Fig. 2 is a case among several variants but nevertheless it is quite representative for our problematic. It is not revolutionary and a large part of its components already exist in more traditional SaaS architectures between companies where a composite service orchestrates services from different service providers. The difference here is that in the cloud context, not only services, but also the computing infrastructure and the BPMS (Business Process Management System) platform can be outsourced and shared. Moreover, in the cloud context, peer-to-peer negotiations are also more difficult and clearly limited: this increases security risks.

For example, the architecture in Fig. 2 is close to this in [4] concerned with SaaS. Nevertheless, some components are impacted and new requested: they are enlightened in white in in Fig. 2 and discussed below from the cloud consumer, cloud broker and cloud provider sides.

1) Cloud consumer side: A cloud consumer, i.e. a client of cloud resources, initiates the deployment of a business process by developing a first version of the BP model including the initial business process logic and some non-functional requirements, i.e. QoS requirements, and especially in our case, security requirements.

In our approach, QoS/Security requirements are formalized in terms of constraints ([5]) which feed the cloud selection algorithm of the cloud broker. Typically, we can note (in addition to other QoS constraints): the pre-assignment of a task to a specific cloud recognized for its expertise in a task domain, or the definition of tasks co-location or separation constraints for organizational purposes, for example the grouping of tasks requiring the same nature of resources, or more specifically related to security, a constraint for separating two tasks in an obfuscation objective, or the requirement of a minimal trust level for a specific task ...

Thus, as depicted in Fig. 2 the consumer tool-kit includes a traditional modelling tool (for example a BPMN[1] editor), a tool for supporting security/QoS requirement elicitation, and finally **a tool called** *obfuscator* **for supporting the obfuscation of a process model**.

[1]BPMN: Business Process Management Notation, ...

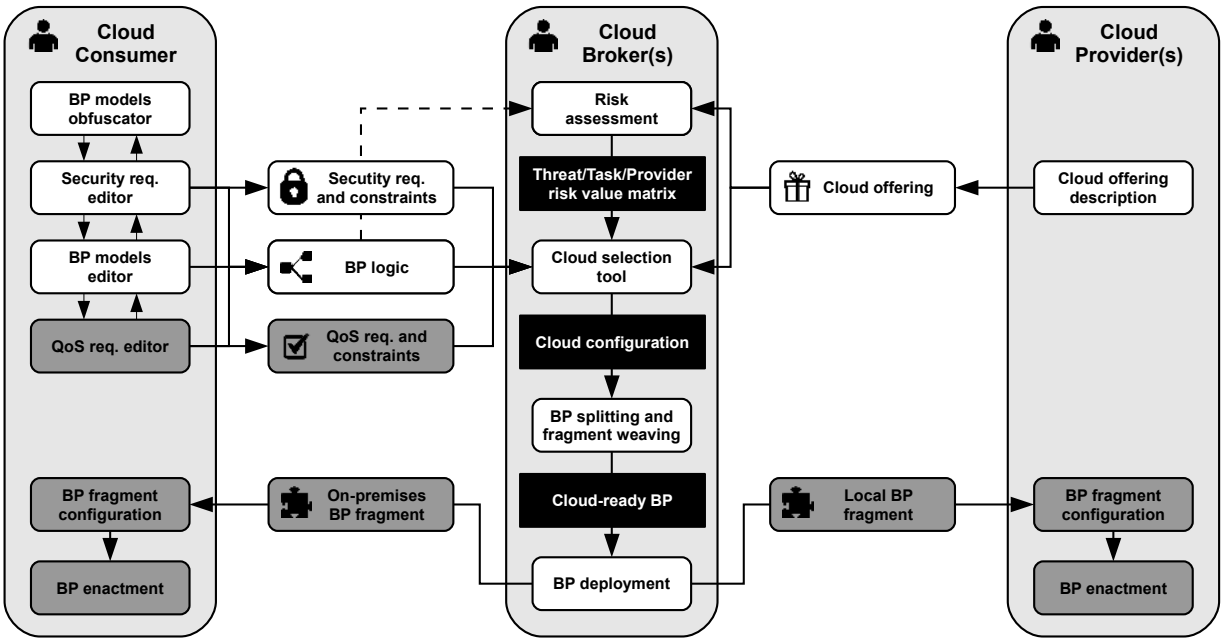

Fig. 2. Architecture

Finally the consumer needs to enact a process, what is more or less complex depending on the contract between the client and the broker, but is not really impacted by our context.

2) Cloud broker side: The central operation of the cloud broker is, based on the BP logic and the security and QoS needs, to assign BP fragments to cloud providers. This can be seen as a multi-criteria optimization problem: "how to find an optimal assignment of clouds to tasks which maximize performance and minimize costs while assuming the requested level of security?" This algorithm inputs directly some QoS/security constraints, and some more elaborated values as a synthetic security risk calculated by a risk assessment tool. **In fact, the result of the obfuscation tool introduced in this paper is a set of constraints which will enrich the QoS/security constraints in input of the global optimization problem resolved by the cloud broker.**

This risk assessment tool confronts the consumer security needs with the cloud providers security promises for calculating, in a first time for each couple task/provider, and in a second time for each potential configuration of clouds, a synthetic risk value. In our work, client requests and cloud promises are compared using a common reference based on the taxonomy work done by the CSA [6] and the ENISA [7] public organizations. See [8] for more on this topic.

When tasks are assigned to cloud providers, the cloud broker projects the BP logic on each selected cloud provider defining in such a way a BP fragment per cloud. The next function is to re-create the BP in its entirety by weaving the resulting BP fragments.

Finally the cloud broker deploys the BP fragments to the concerned providers and establishes the communication links between the different providers, and between the providers and the client, as requested.

3) Cloud provider side: The first function of a cloud provider is its ability to describe the services it offers. Force is to note that currently a service description mainly focuses on the functional dimensions of the service and that the terms of contracts between the cloud and its consumers are generally limited to cost and performance considerations, and that poor care is given to security issues. However, some grounding work still exists to support security properties descriptions of offers: in our work we have used the Security Trust Assurance Registry (STAR) of the Cloud Security Alliance (CSA) [9] which seems to us being a good starting point. Another point concerns the metrics for comparing such cloud descriptions. There is also a lack of standards in this area and in our work we were inspired by the Common Assurance Maturity Model [10] and the EuroCloud Star Audit model [11].

To support the execution of a BP fragment model, a cloud must provide an API that exposes the requested function for executing and connecting BP process fragments.

III. Critical BP fragments characterization

This section is organized as follows. The next section overviews **our approach, i.e. hiding decisions and syntheses by distributing the concerned fragments in different clouds**. Section III-B establishes intuitively some syntactic links between critical fragments and BP models, taking the BPMN notation as a reference (section II-A). Section III-C formalizes **the notions of a decision and of a synthesis** which are the cornerstones of our approach.

A. Approach: decisions and syntheses hiding

Among the different means listed in section II-B, we concentrate here on the objective to split a BP logic into several BP fragments and to assign BP fragments to different cloud providers so that a cloud provider alone will not be able to rebuild valuable information about the BP know-how. In this objective, the first step is to find out the critical business process fragments. Our working hypothesis is that

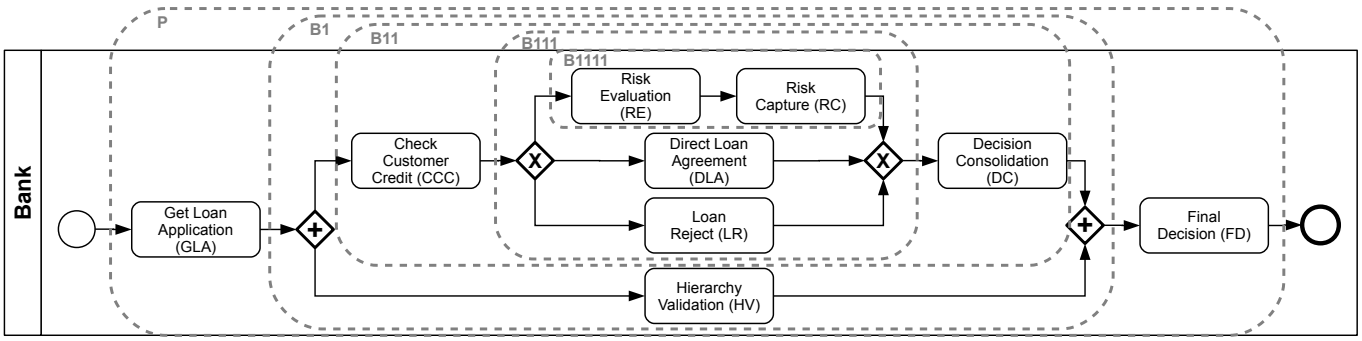

Fig. 3. Splitting the Loan Application process in blocks.

the more critical fragments of a BP model, i.e. fragments which includes more know-how about the process, are these which are concerned with, on the one hand *decisions*, i.e. the places where some strategic choices are made, and on the other hand, *syntheses*, i.e. the places where different contributions are synthesized.

The main problem here is that, while designers are intuitively able to define such fragments, using their knowledge of the process, automating this task, which means to be able to syntactically characterize such critical fragments, is largely more hazardous. Especially, if decisions and syntheses are explicitly contained in some fragments, they can also be implicitly mined from others not apparently so critical. This is typically the case, as developed below, for the different alternative flows associated to a decision, which can allow mining the corresponding decision.

These critical fragments identified, the objective is to split them in fragments to be assigned to different providers while assuming that their combination will preserve the whole BP semantic and continue to hide process decisions and syntheses.

B. Intuition of decisions and syntheses in BP models

The question addressed here is "how decisions and syntheses can be syntactically characterized in a BP model?" The following sections formalize the intuitions which emerged in the study of several academic examples and were globally validated against other such examples.

a) Hypotheses: In the following, we take the well-known BPMN notations as a reference for BP modelling. We make also the hypothesis that our BPMN processes are well structured [12]. To make short, to each opening (x)or-split gateway corresponds a closing (x)or-join gateway, and to each opening and-split gateway corresponds a closing and-join gateway. In addition, the fragments between such brackets do not overlap. It has been demonstrated that it is not a limit in theory and we discuss quickly practical limits in section VI.

Our motivating example in Fig. 1 is well structured because the *and-split* is closed by an *and-join*, the *xor-split* is closed by an *xor-join*, there is no gateway between these two gateways, and the *xor-split-xor-join* block is completely included in the *and-split-and-join* block.

b) Decisions (intuitive definition): We have established a relation between decisions and *(x)or-split* gateways triggering alternatives fragments, and more precisely that the decisions are taken in the task or fragment preceding such gateways. We call *Decision opening* such a critical fragment. In our example, the *CCC* task implements a *Decision opening* fragment.

c) Syntheses (intuitive definition): Respectively, we have associated syntheses to *and-join* gateways synchronizing several flows executing in parallel, and more precisely that syntheses are done in the task or fragment succeeding such gateways. We call *Synthesis closing* such a critical fragment. In our example, the *FD* task implements a *Synthesis closing* fragment.

d) Decision dependency: Also, we find out that if a decision is mainly implemented in a task or fragment preceding an *(x)or-split* gateway, such a decision is often complemented by an action in the task or fragment following the *(x)or-join* gateway closing the opening *or-split*. We say that there is a *Decision dependency* between these two complementing fragments. In our example, there is a *Decision dependency* between *DC* and *CCC*.

e) Synthesis dependency: Respectively if a synthesis is mainly implemented in a task or fragment succeeding an *and-join* gateway, such a synthesis is often prepared in the task or fragment preceding the opening *and-split* gateway corresponding to the closing *and-join* gateway). We say that there is a *Synthesis dependency* between these two complementing fragments. In our example, there is a *Synthesis dependency* between *GLA* and *FD*.

f) Alternative relationship: As introduced above, decisions are also implicitly existing in the flows arising from decisions fragments. In fact, regarding decisions, analysing the alternative flows following an *or-split* can allow mining the decision strategy by comparing the different data states and the conditions flowing in the different alternatives. We say that there is an *Alternatives relationship* between the fragments in these alternative flows. Back to our example, analysing the three flows issued from the *or-split* gateway can allow understanding the algorithm used in the *CCC* task by comparing the different loan properties considered in the three flows. There is an *Alternatives relationship* between these three flows.

Surely things are more difficult and this way of doing is incomplete; for example we had a reflection about the opportunity to separate or not the flows contributing to a synthesis, but this appeared us not so useful and currently we do not consider this case. Also the result of obfuscation can lead to inconsistencies with designer choices (see sec-

tion IV-D). Nevertheless we find that our approach is quite plausible in general and applies in most cases. However, due to this uncertainty the obfuscation service currently remains interactive.

C. Decisions and syntheses specification

The above section has intuitively introduced and characterized the critical fragments associated to decisions and syntheses. This section explains how to specify more formally these notions.

In this objective, we use an internal representation of a BP based on the R-PST model (Refined Process Structure Tree) defined in [12]. In such a structure, decisions and syntheses elements are characterized as nodes with a specific position in the tree, and as some relationships between the corresponding nodes.

Before formalizing decisions and syntheses, we overview the R-PST internal representation of a BP.

1) R-PST representation of a BP: To implement our approach, we have extended the algorithm described in [12] which allows to break down a BP model logic in a R-PST tree. This algorithm has some application conditions consistent with the BP modelling hypotheses we have introduced above.

In a first step, the process is broken down in canonical blocks. A canonical block is a BP fragment with one entry and one exit (SESE: Single Entry/Single Exit). The entry of the canonical block is either a task (in the case of a sequence flow), or a gateway splitting the flow in several alternative sub-flows (*(x)or-split* gateway) or concurrent sub-flows (*and* gateway). The exit of a canonical block started with an *(x)or-split* gateway is an *(x)or-join* gateway (decision block); the exit of a canonical block started with an *and-split* gateway is an *and-join* gateway (parallel block). Roughly speaking, a canonical block is a BP fragment between two complementary *split* and *join* gateways.

A canonical block can include one or more canonical blocks, but either in their entirety or not at all, i.e. blocks cannot overlap. Block imbrication is the hierarchy property used in R-PST.

Fig. 3 depicts the decomposition of the *Loan Application* process in canonical blocks.

More formally, a R-PST is basically a classical tree (see Fig. 4-(a)), i.e. a set of nodes, each node having either one parent or none (the root). Each node has one or several children, or none (a leave). In our model, the set of children is ordered. Thus a node has one *left_brother* (which precedes the node in the list of its parent children) and a *right_brother* (which succeeds it in this list).

2) R-PST tree decorated with decision and synthesis properties: Working with such a structure largely simplifies the characterization and visualization of decision and synthesis elements as illustrated below. In the following, we simply use the well-known technique of attributed grammar [13] to decorate the R-PST tree. Fig. 4-(b), depicts our motivating example, with attributes for representing *Decision opening*, *Synthesis closing*, *Decision dependency*, *Synthesis dependency* and *Alternatives dependency*, as formally defined below:

a) Node type.: Each node is labelled with a type:

$$type : \mathbf{node} \rightarrow \begin{cases} \text{“}X\text{”}, \text{if it is a decision block} \\ \text{“}+\text{”}, \text{if it is a synthesis block} \\ \text{“}seq\text{”}, \text{if it is a sequence block} \\ \text{nothing else} \end{cases}$$

b) BP fragment criticality: The criticality attribute expresses if a fragment is a *Decision opening* or a *Synthesis closing* fragment.

$$criticality : \mathbf{node} \rightarrow \begin{cases} \text{“}do\text{”}, \text{if it is a decision opening} \\ \text{“}sc\text{”}, \text{if it is a synthesis closing} \\ \text{nothing else} \end{cases}$$

Its value is defined as follows:

Decision opening fragment. In the R-PST tree, a node which is the *left_brother* of a node with type = “X” is a decision opening fragment. Its criticality attribute has the value “do”:
Let n be a node of the R-PST tree,
$type(n) = \text{“}X\text{”} \rightarrow criticality(left\_brother(n)) = \text{“}do\text{”}$
To visualize this property, the *CCC* task is exposed with “do” in Fig. 4-(c).

Synthesis closing fragment. In the R-PST tree, a node which is the *right_brother* of a node with type = “+” is a synthesis closing. Its criticality attribute has the value “sc”.
Let n be a node of the R-PST tree,
$type(n) = \text{“}+\text{”} \rightarrow criticality(right\_brother(n)) = \text{“}sc\text{”}$
To visualize this property, the *DC* task is exposed with “sc” in Fig. 4-(c).

c) Decision dependency:
In a R-PST tree, there is a *decision_dependency* between the *left_brother* and the *right_brother* of a node with type “X”. Let n be a node of the R-PST tree,

$$type(n) = \text{“}X\text{”} \rightarrow decisions\_dependency(left\_brother(n), right\_brother(n))$$

To visualize this property, a decision dependency is depicted between fragments *CCC* and *DC* in Fig. 4-(c).

d) Synthesis dependency:
In a R-PST tree, there is a *synthesis_dependency* between the left_brother and the right_brother of a node with type “+”. Let n be a node of the R-PST tree,

$$type(n) = \text{“}+\text{”} \rightarrow synthesis\_dependency(left\_brother(n), right\_brother(n))$$

To visualize this property, a synthesis dependency is depicted between fragments *GLA* and *FD* in Fig. 4-(c).

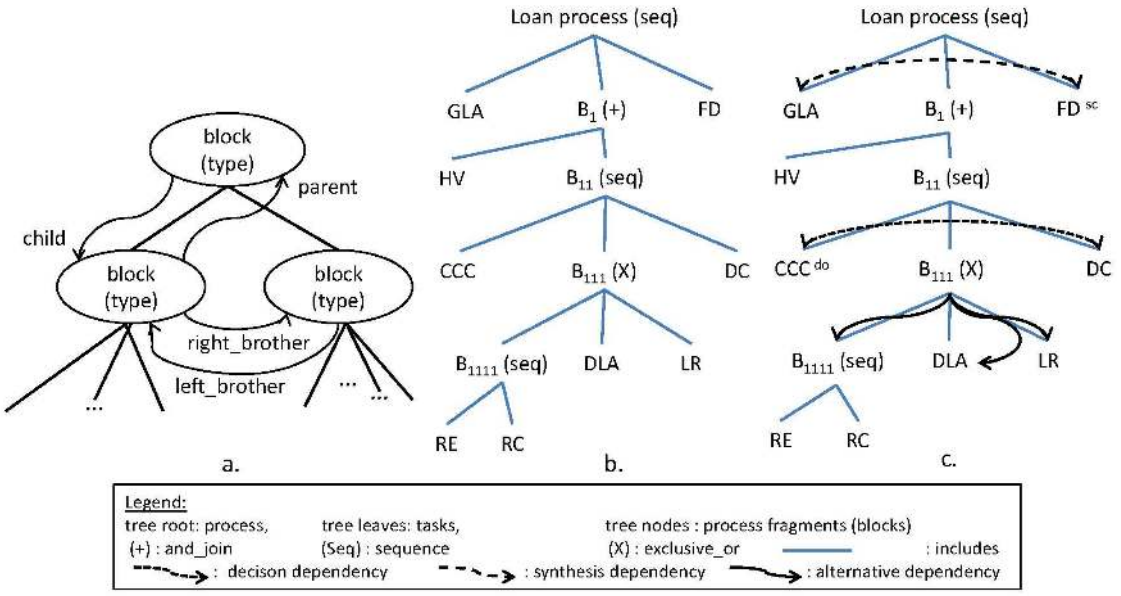

Fig. 4. (a): R-PST abstract concepts, (b): Loan process R-PST, (c): decorated Loan process R-PST

e) Alternative dependency:

In a R-PST tree, there is an alternative dependency between all the children of a node which type is "X". Let n be a node of the R-PST tree,

$$type(n) = \text{“}X\text{”} \rightarrow alternative\_dependency(children(n))$$

To visualize this property, an alternative relationship is depicted between fragments B_{1111}, *DLA* and *LR* in figure 4-(c).

IV. Semi-automatic BP logic obfuscation

In this section we explain how the critical fragments are used to obfuscate a BP model.

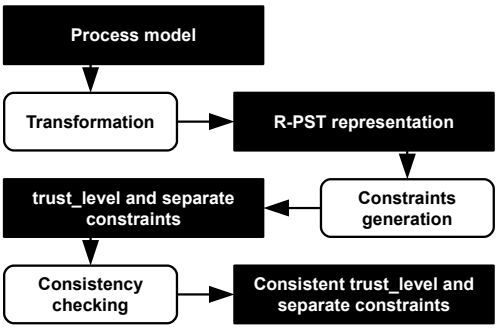

Fig. 5. Obfuscator architecture

A. Approach

As introduced above, this work has been developed in a broader context where obfuscation is just one dimension: risk management [8], cost and performance optimization [14] are other ingredients that we consider in a global optimization approach. As a consequence, the result of the obfuscation service is not directly the assignment of fragments to cloud providers, but the definition of *trust-level* and *separation* constraints as defined below, which are inputs for this optimization algorithm. Also, in Fig. 2, trust-level and separation constraints are parts of the *security requirements and constraints* provided by the cloud client and used by the *Risk assessment* and *Cloud selection* components of the cloud broker.

Trust-level constraints allow assigning to a *decision opening* fragment (respectively to a *synthesis closing* fragment) a minimum trust level which will be used to assert that the cloud assigned to this fragment by the optimization algorithm is trusted enough. Separation constraints requests two fragments not to be assigned to the same cloud.

This lead to the following process for obfuscation constraints generation (Fig. 5):

1) In the first step, we build the hierarchical representation of the BP logic as a R-PST tree (section III-C1).
2) In the second step, we decorate the tree with know-how attributes (as defined in section III-C2).
3) In the third step, we apply rules **1** to **5** below on the tree structure using know-how attributes to generate separation and trust-level constraints.
4) In the fourth step, the consistency of constraints is checked against constraints of other sources coming from designer choices or cost, performance optimization.

B. Rules for BP fragment assignment for decision and synthesis hiding

Consistently with the general principles introduced above, we have defined the following rules for assigning BP fragments to cloud providers:

- **Rule 1** (concerned with *Decision opening*): if possible, break down the fragment preceding the *or* gateway, or at least assign it to a cloud with a high level of trust
- **Rule 2** (concerned with *Synthesis closing*): if possible break down the fragment succeeding the *and* gateway, or at least assign it to a cloud with a high level of trust
- **Rule 3** (concerned with *Decision dependency*): assign the fragment preceding an opening *or-split* gateway and the fragment following the corresponding closing *or-join* gateway to two different clouds

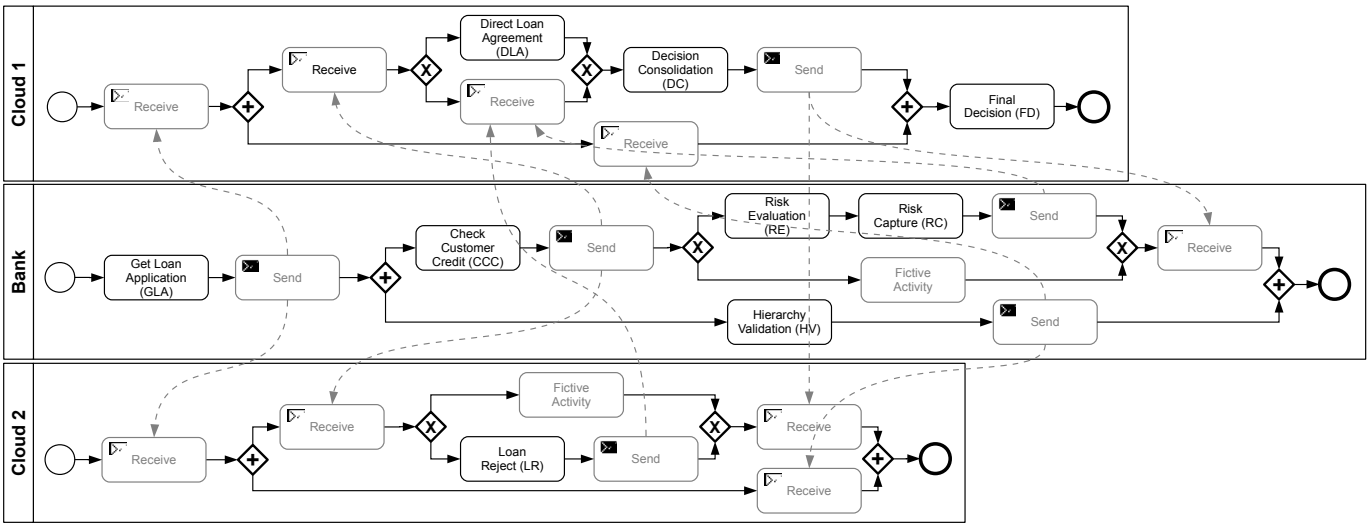

Fig. 6. The fragmented Loan Process

- **Rule 4** (concerned with *Synthesis dependency*): assign fragment preceding an opening *and-split* gateway and the fragment following the corresponding closing *and-join* gateway to two different clouds
- **Rule 5** (concerned with *Alternatives relationship*): assign the different fragments following a *or-split* gateway (the different alternatives of the choice) to different clouds.

Regarding rule **1** and rule **2**, splitting decisions and syntheses comes from free when *Decision opening* and *Synthesis closing* are themselves complex fragments including at least one *or* or *and* blocks. In the case it is a single task, only a recommendation of splitting the fragment can be given in addition to assign it to a cloud with a good trust level.

C. Constraints generation

A trust level constraint $trust\_level(f_i) > l$ is assigned to each fragment (block in the R-PST tree) which critical attribute value is *do* or *sc*. It is either a *decision opening* (rule **1**) or a *synthesis closing* (rule **2**) fragment. This level l is application dependent. Nevertheless it requires a global reference for all cloud provider; see [8] for a discussion about the definition of such a global reference.

A $separate(f_i, f_j)$ constraint is defined for each $decision\_dependency(f_i, f_j)$ (rule **3**) and each $synthesis\_dependency(f_i, f_j)$ (rule **4**).

A $separate(f_i, f_j)$ constraint is defined (rule **5**) for each couple of fragments f_i, f_j taken from an $alternative\_relationship$.

D. Interactive consistency checking

The objective of this activity is to verify the consistency of constraints, and especially the constraints generated by the obfuscator tool against the constraints of other sources. In theory, due to our modelling hypotheses and especially the SESE (Single Entry/Single Exit) principle, inconsistencies between constraints automatically generated by the obfuscator cannot occur.

Especially, the consistency of the constraints automatically generated and the constraints manually defined by the designers have to be verified. This is due, on the one hand to the uncertainty in the automation of decisions and syntheses detection, and on the other hand to the strategic decision of designers, for example the decision to retain on premises all or a part of the components of a critical fragment.

Regarding the $trust\_level$ constraints, there is an inconsistency when the obfuscator tool and a designer propose different trust levels for the same fragment. In such a case, the final choice is on the designer responsibility.

Regarding the $separate$ constraints, an inconsistency exists when a $separate(f_i, f_j)$ constraint, generated by the obfuscator, and a $colocate(f_i, f_j)$ constraint, defined by a designer, occur simultaneously.

A $colocate(f_i, f_j)$ constraint means that fragments f_i and f_j must be assigned to the same cloud provider. This can be for security purpose; for example a designer decide to maintain f_i and f_j on premises because they are very critical fragments. This can be also for other reasons, for example, cost or performance optimization (see [5] for more about co-location constraints).

In the case of such a conflict, the $colocate(f_i, f_j)$ constraint wins in general as it corresponds to a designer choice, but in any case, the final choice rests yet on his/her responsibility.

E. Back to the motivating example

Applying this algorithm to our example, 7 constraints are generated:

$trust\_level(FD) > l_1$
$trust\_level(CCC) > l_2$
$separate(GLA, FD)$
$separate(CCC, DC)$
$separate(B_{1111}, DLA)$
$separate(B_{1111}, LR)$
$separate(DLA, LR)$

Fig. 6 depicts a distribution of the loan process consistent with these constraints.

Just for the example, a designer can decide to maintain on premises *CCC* and *DC* thus generating the constraint $colocate$(*CCC*,*DC*) inconsistent with $separate$(*CCC*,*DC*). In consequence, he has to delete the $separate$(*CCC*,*DC*) constraint.

V. State of the art

As introduced above, this work is developed in the vein of [15] and of our previous work [2]. But these propositions remained at the level of principles and methodologies.

Regarding works directly addressing privacy preservation, force is to note that most of them are concerned with the execution time ([16] for example) and that very few apply at the design time. At design time, [17] generate obfuscated BP models from BP, but with a different purpose (model sharing and analysis) and techniques which are not adapted to our context. Between design time and execution time, [18] is concerned with privacy preserving of BP models provenance, what is complementary, but not directly related to our work.

Other works are concerned with process model splitting and process fragments weaving [19], [20], [4], [21] but other work either do not consider the privacy dimension at all, or do not automatically generate splitting recommendations from a direct analysis of the BP logic.

VI. Conclusion

Preserving the know-how implemented in its processes is the first guarantee to provide before an enterprise can accept to deploy them in the cloud. As a contribution to this topic, this paper has introduced a first semi-automatic approach for obfuscating a business process by simply analysing the BP logic for efficiently splitting it in several fragments, and assigning each fragment to different clouds, so that a cloud alone cannot discover valuable information about the enterprise know-how.

One can consider that the five rules above can generate an important fragmentation of process models. But, on the one hand this can be the price to pay for the preservation of know-how, and on the other hand this fragmentation can be reduced by designer decisions who can decide to maintain on premises several critical fragments or that some fragments are finally not so critical and can be combined. This can also be optimized on the basis of measures of know-how risk and the acceptance of a threshold for risk [8] (out of the scope of this paper). The fact that we consider well-parametrized process models is also a restriction, but the same hypothesis is often done in most theoretical work about BP. Also, we consider that in our context, we are mainly concerned with high level coordination, and at this level, this constraint seems realistic.

While a complete validation of the choices made in our algorithm yet remains to be done against more numerous and realistic process models, our experiments against academic examples has confirmed our intuitions and the plausibility of the approach. We have also demonstrated the technical feasibility of the approach by simply extending a BPMN to R-PST algorithm.

Regarding future work, if the objective of this work is to prevent one cloud provider to discover some know-how of a client company, it does not consider how an alliance of several cloud providers can attack our solution: it is what we are currently investigating.

References

[1] C. Collberg and C. Thomborson, "Watermarking, tamper-proofing, and obfuscation tools for software protection," *Software Engineering, IEEE Transactions on*, vol. 28, no. 8, pp. 735–746, Aug 2002.

[2] E. G. an Nicolas Mayer and C. Godart, "A general approach for a trusted deployment of a business process in clouds," in *ACM Fifth International Conference on Management of Emergent Digital EcoSystems (MEDES)*, 2013, pp. 92–99.

[3] W. M. P. van der Aalst, B. F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A. J. M. M. Weijters, "Workflow mining: A survey of issues and approaches," *Data Knowl. Eng.*, vol. 47, no. 2, pp. 237–267, 2003.

[4] W. Fdhila, M. Dumas, C. Godart, and L. García-Bañuelos, "Heuristics for composite web service decentralization," *Software and System Modeling*, vol. 13, no. 2, pp. 599–619, 2014.

[5] W. Fdhila, M. Dumas, and C. Godart, "Optimized decentralization of composite web services," in *CollaborateCom'10*, 2010, pp. 1–10.

[6] Cloud Security Alliance, "The Notorious Nine - Cloud Computing Top Threats in 2013," Tech. Rep., 2013.

[7] European Network and Information Security Agency, "Benefits, risks and recommendations for information security," Tech. Rep., 2009.

[8] E. Goettelmann, K. Dahman, B. Gâteau, E. Dubois, and C. Godart, "A security risk assessment model for business process deployment in the cloud," in *IEEE International Conference on Services Computing (SCC)*, 2014, pp. 307–314.

[9] Cloud Security Alliance, "Security, Trust and Assurance Registry," https://cloudsecurityalliance.org/star/, Tech. Rep., 2014.

[10] Common Assurance Maturity Model, "Common Assurance Maturity Model Guiding Principles," http://www.common-assurance.com/resources/Common-Assurance-Maturity-Model-vision.pdf, 2010.

[11] EuroCloud Deutschland eco e.V., "Eurocloud Star Audit," http://www.saas-audit.de/en/511/requirements/, 2012.

[12] A. Polyvyanyy, L. García-Bañuelos, and M. Dumas, "Structuring acyclic process models," in *Business Process Management - 8th International Conference (BPM)*, 2010, pp. 276–293.

[13] D. E. Knuth, "The genesis of attribute grammars," in *Attribute Grammars and their Applications, International Conference (WAGA)*.

[14] K. Bessai, S. Youcef, A. Oulamara, C. Godart, and S. Nurcan, "Scheduling strategies for business process applications in cloud environments," *IJGHPC*, vol. 5, no. 4, pp. 65–78, 2013.

[15] M. Jensen, J. Schwenk, J. Bohli, N. Gruschka, and L. Iacono, "Security prospects through cloud computing by adopting multiple clouds," in *CLOUD'11*, 2011, pp. 565–572.

[16] R. Conforti, M. L. Rosa, G. Fortino, A. H. M. ter Hofstede, J. Recker, and M. Adams, "Real-time risk monitoring in business processes: A sensor-based approach," *Journal of Systems and Software*, vol. 86, no. 11, pp. 2939–2965, 2013.

[17] H.-G. Fill, "Using obfuscating transformations for supporting the sharing and analysis of conceptual models," in *Multikonferenz Wirtschaftsinformatik*, 2012.

[18] M. Bentounsi, S. Benbernou, and M. J. Atallah, "Privacy-preserving business process outsourcing," in *IEEE 19th International Conference on Web Services (ICWS)*, 2012, pp. 662–663.

[19] R. Khalaf and F. Leymann, "E role-based decomposition of business processes using BPEL," in *IEEE International Conference on Web Services (ICWS)*, 2006, pp. 770–780.

[20] E. Goettelmann, W. Fdhila, and C. Godart, "Partitioning and cloud deployment of composite web services under security constraints," in *IEEE International Conference on Cloud Engineering (IC2E)*, 2013, pp. 193–200.

[21] E. F. Duipmans, L. F. Pires, and L. O. B. da Silva Santos, "A transformation-based approach to business process management in the cloud," *J. Grid Comput.*, vol. 12, no. 2, pp. 191–219, 2014.