Assessment and Impact Analysis for Aligning Business Processes and Software Systems
Lerina Aversano, Thierry Bodhuin and Maria Tortorella
RCOST - Research Center On Software Technology
Department of Engineering, University of Sannio
Via Traiano, Palazzo ex-Poste, 82100, Benevento, Italy
Tel: +39 0824 30 5551 / 5553 / 5554
aversano/bodhuin/tortorella@unisannio.it

ABSTRACT
Business processes and existing software systems must be aligned so that software systems can adequately support the business processes in order to be effectively used within them. The alignment characteristic needs to be considered even during the execution of an evolution process. In particular, a strict relationship exists between the evolution of a legacy system and that of the supported business process. Therefore, the requirements for evolving a software system embedded in a business process are to be defined on the basis of the change needing to be performed on the process activities. In fact, any modification performed in the business process activities and/or supporting software system may impact the process activities in terms of input/output and/or purpose of the software system and, therefore, cause misalignment. A coarse grained strategy is proposed for detecting misalignment between software systems and supported business processes when a change is executed. In addition, the strategy supports the identification of all the objects, either software system components or process activities, affected by a change and needing to be considered during the evolution process, for keeping the alignment and ensuring the technological support to the business process. The strategy proposes the exploitation of quality parameters, for codifying the alignment concept, and impact analysis techniques, for propagating the change and identifying all the objects affected by a change and requiring new evolution interventions.

Keywords
Evolution, Business Process Re-engineering, Evaluation, Impact Analysis

1. INTOXOINTRODUCTION
Fast changes of business requirements are forcing enterprises to innovate their business processes [6, 8] as well as their supporting software systems, for considering arising and/or changing requirements. Innovation activities have to consider the alignment between the business processes and the supporting software systems.

A view of business and technological alignment defines it as the degree to which the information technology mission, objectives, and plans, support and are supported by the business mission, objectives, and plans [12]. Moreover, it involves “fit” and “integration” among business strategy, IT strategy, business infrastructure, and IT infrastructure [7, 10]. Unfortunately, modifications in an operational context can cause misalignments between business processes and supporting software systems. They can be due to either technological and/or management innovations, or unchecked change in the way the activities are executed or the supporting software systems are exploited. Furthermore, a modification may usually not only regard the considered object but it can impact also other objects having a dependence relation with the modified one.

The evolution activities should take in consideration the alignment between the business processes and their support instrumentation. Misalignment must be detected and alignment actions must be identified and executed. The alignment consists of modification interventions that involve one or more objects of the analyzed business process, which are mainly activities and components of the supporting software systems. The modification impacts on the other objects must be detected and, on the basis of the relations between those objects, changes must be planned. For example, in order to avoid misalignment, a change of an activity may require modifications in the software system components supporting it and/or in the dependent activities. For the same reason, modifications on a software system component may require the analysis and modifications of the activities it supports and/or other software components.

This paper presents a strategy to be applied during evolution for detecting a misalignment between business processes and supporting software systems and identifying the process objects to be changed for restoring the alignment. The strategy considers a set of attributes, that represent indicators of a possible misalignment, and an approach based on impact analysis [4,11], for identifying all the process components impacted by an initial set of changes.

Section 2 describes the proposed alignment strategy and introduces the used technologies. The subsequent section discusses the techniques used for supporting the application of the strategy. Section 4 presents a case study, and final remarks are given in the last section.

2 ALIGNMENT STRATEGY
One of the key problems to be faced for analyzing, measuring and effectively evolving a business process and related supporting systems by keeping their alignment, is modeling what has to
be analyzed and keeping the model updated during the evolution activities. Any formalism can be effective provided that it is complete and easy to be used and understood. In addition, it is important that it respects standard requirements to facilitate the diffusion and comprehension of the constructed process models and the cooperation among different actors for their updating if evolution is needed. The formalism used in the presented case study is the Unified Modeling Language (UML) [9], chosen for its simplicity and standard characteristics, even if it needs to be opportune extended for supporting the process modeling.

**Figure 1. Activities of the alignment strategy**

Figure 1 illustrates the main activities included in the proposed alignment strategy. A coarse grained description of the four activities follows:

**Change execution**: regarding the introduction of modifications in an analyzed business process. It includes also the case of the development of a new business process, considered as the introduction of a major innovation. A change may consist of: a managerial innovation, regarding the way the business process is executed and, then, its activities; or a technological innovation, regarding modification in the software system components.

**Misalignment detection**: concerning the identification of a misalignment due to the performed change. The main problem faced in this task regards how to measure and check the alignment level. The solution adopted in this paper consists of the evaluation of a set of parameters and the assignment of thresholds to them. Evaluating the parameters and performing a gap analysis of the obtained measures with the given thresholds, permit to detect the misalignment between the business processes and current technological solution. If the gap is negative, it means that the alignment codified in the thresholds does not exist and actions are needed. This activity should be executed periodically, even when innovations have not been officially introduced but a change can be due to an incorrect execution of an activity and exploitation of the supporting software system.

**Needed change identification**: if the previous phase detects a misalignment, corrective actions are required. They have to involve the business process activities and software system components having a dependence relation with the objects modified in the first phase and being impacted by the performed change. In the current research, the identification of the process objects to be considered in the alignment actions is obtained by considering the dependence relationships existing among the process objects, and propagating the change among them by applying impact analysis techniques.

**Change planning**: once identified the objects to be involved in the alignment actions, the innovation activities to be performed have to be identified and planned as they cannot be applied all in the same moment but on the basis of some priorities. To fulfill this purpose, some approaches have been defined in literature based on measurement activities and critiquing techniques [1, 2], and they can be applied for supporting this activity.

This paper concerns mainly the second and third phases of the strategy. In particular, the following two subsections outline the evaluation of the alignment, supporting the second phase, and the impact analysis technique applied in the business context, helping to identify the needed change.

### 3 ADOPTED TECHNIQUES

This section describes the techniques applied for supporting the strategy above. In particular, the first technique regards the evaluation of a set of parameters, with the aim of assessing the alignment between a business process and the supporting software system. While, the second technique is based on impact analysis [4, 11] and aims at identifying activities and software system components affected by a performed modification.

#### 3.1 Detecting the misalignment

To be able to detect the misalignment, it is needed to codify the alignment and define how to evaluate it. This paper considers the assessment of a set of parameters codifying the alignment between business processes and supporting software systems.

A first set of parameters are:

**Technological Coverage (TC)**, indicating the percentage of process activities adequately supported by a software system; it can be evaluated as a percentage of the automatically supported activities on the total number of activities.

**Technological Adequacy** for each activity. In particular, the Technological Adequacy of activity $i$ ($TA_i$) indicates how adequate is the used software system for supporting activity $i$. The evaluation of this parameter considers the following aspects: the technological coverures of the analyzed activity, indicating how many activity tasks are automatically supported; the input the activity requires and that are elaborated as input by the supporting software system; the output the activity produces and that are produced as results by the supporting software system; the time the software system spends for elaborating the expected results and its compliance with the time required by the activity execution; the knowledge and resources the software system requires for being executed in order to support the activity.
In general terms, the adequacy is high if the software components analyze the input of activity \( i \) and produce the output expected by the execution of the activity in the expected time and using the available resources. Other characteristics can be considered on the basis of the activity complexity.

The evaluation of the parameters and the assessment of the listed characteristics can be performed through the definition of a measurement framework, on the basis of the Goal Question Metrics (GQM) paradigm, defined by Victor Basili [5]. The use of this paradigm starts from the definition of the measurement goals, and analyzes the questions to be answered for reaching the stated goals and the metrics to be assessed for answering the questions.

In the analyzed context, the measurement goal can be formulated as: *Analysis of the alignment degree existing between a business process and the supporting software systems from the point of view of the organization’s managers and the business process’s executors.*

The main questions to be answered for achieving this goal are:

- **which is the Technological Coverage (TC) of the business process?**
- **which is the Technological Adequacy of activity \( i \) (TA\(_i\)) of the business process, for each activity?**

Other questions can be formulated on the basis of the business process complexity and characteristics of the activities.

The metrics to be considered for answering the questions regard the assessment of the information listed above and additional characteristics identified on the basis of needs and matured experiences. For reason of space, it is not possible to present here in a greater detail the measurement framework. In any case, the assessed metrics and their aggregation permits the evaluation of TC and TA\(_i\), for each activity \( i \).

The alignment between business process and software system can be codified through the introduction of the thresholds Thre\(_TC\) and Thre\(_TA_i\), for each activity \( i \). These thresholds represent the minimum wished alignment values. It is worthwhile underline that the threshold values depend on the analyzed business process and process activities. In fact, if the business process includes activities requiring the execution of completely manual tasks, the thresholds values assume a value less than 100%.

Once the threshold values, Thre\(_TC\) and Thre\(_TA_i\), are defined, the considered business process and supporting software systems are aligned if:

\[
\text{Thre}_{TC} \leq \text{TC} \quad \text{and} \quad \text{Thre}_{TA_i} \leq TA_i \quad \text{for all activity } i.
\]

Once defined the alignment in the terms specified above, a change can cause Thre\(_TC > TC\). The change can be due to a management (introduction and/or modification of an activity) or technological (modification of the software system) innovation and it decreases the technological coverage potentially causing misalignment. Analogously, the change performed, for example, on activity \( i \) can cause the decreasing of the technological adequacy for that activity, that is Thre\(_TA_i > TA_i\). This indicates misalignment in activity \( i \), which must be detected and repairing actions must be identified and performed for reestablishing the alignment. It can also happen that a change does not decrease the values of the parameters, preserving the alignment. This can happen if, for example, a not automatically supported activity is deleted or technological innovations increases both TC and TA\(_i\), for all \( i \).

### 3.2 Propagating the change

Impact analysis is a very widespread technique in the field of software maintenance. In that context, it has been defined as: *The task of assessing the effects of making a set of changes to a software system* [11]. Its main objective is to define the complete set of actions to be performed on all the considered components of the system. The considered components may include source code and any operational documentation and procedures.

In the business process management field, the impact analysis techniques may represent a useful instrument for identifying the effect of a change to the business process and supporting software systems. To fulfill this purpose, business and software assets have to be considered and the first step to be performed consists of modeling the full system that is under analysis. Therefore, the set of objects to be modeled does not include just the software system and its components, but also the organization, business processes, process activities and so on. In addition, identified all the objects, it is necessary to understand the dependence relations among them. A first list of dependences is as follows:

- **include(process, activity), as a business process is composed of a set of activities;**
- **use(process, software_system), as the business process can be supported by a software system, software_system;**
- **composed_of(software_system, component), as a software system is composed of a set of software components;**
- **depend_on(activity1, activity2), as a business process activity, activity1, depends on another activity, activity2, as, for example, activity1 processes information needed for starting activity2, or activity1 can be executed only after activity2 has been completely performed;**
- **supported_by(activity, component), as an activity, activity, is supported by the software system component component;**
- **depend_on(component1, component2), as a software component, component1 can depend on another software component, component2, in different ways: use, data, inheritance, composition and so on.**

All the considered objects and reciprocal dependence relationships form a graph, called *Dependency Graph*, linking all the objects related by a dependence relationships.

Once the Dependency Graph has been constructed, it is possible to apply impact analysis [4, 11] for identifying all the objects, business process activities and software systems components, that can be affected by a modification performed on an object, and that need to be changed in order to keep business processes and software systems aligned. To effectively perform this task, it is
necessary to define what is intended for propagation of the impact. To fulfill this purpose, it is needed to define:

- the types of modifications, that can be performed on an object; in particular:
  - an activity may be modified in the way it is performed, `exec_act_change`, or in the interaction modality with the other activities, `inter_act_change`;
  - a software system component may be modified in its interface, `inter_comp_change`, or in its implementation, `impl_comp_change`;

- the propagation rule, describing when and how to propagate a change to the connected objects, for example:
  - if an activity is modified in the way it is executed, the change potentially impacts the software system components supporting it; in fact, they may not any more offer an adequate support to the activity;
  - if an activity is modified in the way it interacts with the other activities, it potentially impacts both the activities that depends on/from it and the software system components supporting it;
  - if a software system component is modified in its interface, the change potentially impacts the activities using the software component and the other software system components depending on it;
  - if a software system component is modified in its implementation, this change potentially impacts the software system components depending on/from it.

A propagation rule may be formalized in the following way [4]:

\[(m_1, c_1, r, c_2) \rightarrow m_2\]

which means: if a modification of type `m_1` is applied to an object `0_1` of type `c_1` which is connected to an object `0_2` of type `c_2` through a link of type `r`, then a modification of type `m_2` is to be applied to `0_2`.

Examples of propagation rules are the following:

- `inter_act_change, activity, depend_on, activity` \(\rightarrow\) `exec_act_change`
- `exec_act_change, activity, supported_by, component` \(\rightarrow\) `impl_comp_change`

The impacts on an object is recursively propagated in order to obtain the complete set of changes deriving from a modification. Anyway, it is necessary to consider that some of the propagated impacts are just potential. For example, if an activity is changed in the way it is performed, the change potentially impacts the software system components supporting the activity. But, if the activity tasks that are modified are originally manually executed, no modification is required on the software system. For this reason, the impact propagation activity cannot be completely automatic, and human knowledge support is needed for deciding if a potential impact is rejected or changed in real, for being propagated on the other objects.

In any case, the change requirements should be identified so that the alignment characteristic is kept for the aligned activities and restored for the misaligned ones. Guidelines for identifying the change requirements can be found in [3].

4. CASE STUDY

The proposed strategy has been applied in a case study, concerning the process performed by the traffic police for the management of the fine minute emitted when a driver is violating the Highway Code. An overview of the process activities and the control flow among them is given in Figure 2(a). By analyzing the figure, it is possible to extract the relationships among the activities, `depend_on`. The process uses a software system specifically designed for supporting all the traffic police’s processes. The system is implemented in Java and consists of 6 applications for about 110 KLOC, about 1.550 classes and 6.900 methods.

The process mainly concerns the verifications performed by the traffic police regarding the respect of the existing Highway Code from the drivers. Once a violation comes out by human or automatic check, the related fine minute is filled in. Then, the process continues with the notification and payment phases, or eventually the appeal activity.

The software system has been specifically designed for supporting the management of fine minutes. Therefore, it supports many of the process activities. In particular, the supported activities are indicated in Figure 2(a) by the label `automated`.

If the Technological Coverage, `TC`, is calculated as:

\[TC = \frac{AutAct}{AllAct}\]

where `AutAct` is the number of automatic activities and `AllAct` is the total number of activities, Figure 2(a) shows that `TC` can be evaluated for the analyzed process at 53%.

The value of the Technological Coverage cannot be 100%, as all the activities that are not automated in the analysed process cannot be supported by any software system, as law constraints require that they are manually executed. For this reason, it is possible to assume that the threshold of `TC`, `Thre_TC`, for the analyzed process is equal to 53%, indicating that all the activities that can be automated have to be effectively supported by a software.

As, the software system was specifically implemented for completely supporting the execution of the automated activities, the measurement of the considered metrics, previously described, has permitted to evaluate the Technological Adequacy that assumed a value of 100% for all the activities. This value is also considered the threshold of the Technological Adequacy, `Thre_IA`, for all the automated activities.

The analysed process directly depends by the Highway Code. Therefore, each time it changes, also the process has to change and the software system may be impacted and require software change for keeping the full support.
In the analysed case study, the change that is considered consists of the introduction of a score in the driving license. With these innovations, the driving licence of each driver has a starting score, which may be reduced with violations to the Highway Code. When the score is reduced to 0 the driving license is lost.

The described innovation has an impact on the process. Figure 2(b) presents the changed process, and highlights all the changed and impacted activities. In a greater detail, the first change on the process consists of the analysis of the Highway Code for checking the score to be reduced when a driver makes a violation. This implies changes in activities “Automatic checking of a violation to the rules” and “Human checking of a violation to the rules”. The change in the first cited activity impacts on the supporting software system components and 32 classes are impacted and need to be modified. Another potential impact involves activity “Data retrieval by the Motorizzazione”, but it is not considered as its execution does not have to be modified. The change on activity “Human checking of a violation to the rules” impacts on two following activities, which are “Paper fine minute emission” and “Fine minute emission by notebook”, and propagates on the next activities. Among them, only the impact on activity “Fine minute data entry” is propagated, as the impact on the other activities is just potential and it does not require any change.

The changes in the two impacted activities, “Fine minute emission by notebook” and “Fine minute data entry”, that are supported by the software system, cause a misalignment with the supported software components.

In fact, the activities consider one new input, that is the score, which is not processed by the software components. Likewise, the automatically produced outputs are not consistent with the expected ones, as they do not include the missing information. Therefore, the Technological Adequacies of the two activities decreases their values to an assessment less than the threshold, which is 100%. The restoring of the alignment requires modifications of the supported software components. The analysis revealed that the activities are supported by the same components, as they required the same functionalities. In particular, 24 classes are potentially impacted, and the application of the propagation rules potentially impacts 72 classes, for a total of more than 1,300 methods. This demonstrates that the advantage of the proposed strategy consists of suggesting the potentially impacted objects, whose number is sensibly less than the dimension of the system. In addition, the number of the real impacted classes and methods decreases with the help of the software engineer that may reject some potential impacts.

Finally, the new Highway Code required the Notification to the Motorizzazione (a public administration department managing the driving licenses) of the score reduction. This entailed the introduction of a new activity. As a consequence, the new activity was not supported by the software system, the Technological Coverage of the process reduced to 50%, that is less than 53%, the technological coverage threshold. This caused misalignment and an extension of the software systems functionalities was required for establish again the alignment.
5 CONCLUSION
This paper describes a coarse grained strategy for detecting misalignment between business processes and supporting software systems when changes are executed, and for identifying which objects should be considered by additional change for restoring the alignment. The strategy proposes the exploitation of quality parameters and impact analysis techniques. The parameters codify the alignment through thresholds representing the minimal values their measures can assume. If a parameter measure assumes a value lower than the associated given threshold, the business process and supporting software systems are misaligned and the misalignment needs to be corrected. As the execution of change on a business process causes misalignment in other process activities, applying an impact analysis technique helps propagating the change and identifying all the process objects impacted by the change and requiring new interventions. The paper discusses also a case study that, in favor of brevity, has been only synthetically described. The case study showed that an innovation might bring to other changes that can be of two kinds: modification and introduction of new objects. For example, the software system can be modified in the existing software components or in the implementation of new software components.

Further effort is required for better formalizing the strategy and introduced techniques. Moreover, their adequate usage requires the support of a software environment, as much information has to be gathered and analyzed. As some partial software solutions already exist [1], the realization of the software environment can be made easier by exploiting them. In any case, it has to be kept in mind that the impact propagation process cannot be completely automatic, as human experience is needed for deciding about the change actions to be considered and propagated and those to be rejected. Only in this way, it is possible to stop the propagation; otherwise the risk is that the propagation will involve all the process components.

The research and experiences made showed that techniques applied in other fields very often are effectively applied to other fields. In this paper, techniques developed for software engineering have been applied in organizational engineering.

REFERENCES