Impact Analysis for Supporting the Co-Evolution of Business Processes and Supporting Software Systems

Thierry Bodhuin, Raffaele Esposito, Cristina Pacelli and Maria Tortorella

RCOST - Research Centre On Software Technology,
Department of Engineering, University of Sannio
Via Traiano, Palazzo ex-Poste
82100 Benevento, Italy
{bodhuin, r.esposito, pacelli,tortorella}@unisannio.it

Abstract. The co-evolution of business processes and supporting software systems is needed for keeping them aligned and it requires managerial and technological effort for adequately planning it. In fact, any modification performed in the business process activities and/or supporting software systems 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 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, for codifying the alignment concept, and impact analysis techniques, for propagating the change and identifying all the process objects impacted by the change and requiring new interventions.

1 Introduction

Fast changes of business requirements are forcing enterprises to innovate their business processes [8, 9] as well as their supporting software systems, incorporating new requirements implementation. The evolution activities should take in consideration the alignment between the business processes and their support instrumentation (BPMDS 2003).

Nevertheless, modifications in the 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. Unfortunately, a modification may regard not only the considered object but it can impact also other objects having a dependence relation with the modified one.

Misalignment must be detected and alignment actions must be acted. The alignment can consist of modification interventions that can involve one or more objects of the analyzed business process, which are mainly activities and components of the supporting software systems. Therefore, a modification of an object may impact other objects. Such impacts may be detected and change may be planned on the basis of the relations between those objects. For example, in order to avoid misalignment, change of an activity may require modifications in the dependent activities and/or in the software system components supporting the activity. For the same reason, modifications on a software system component may require the analysis and modifications of the other software components and/or activities supported by the software system component.

This paper introduces a strategy to be applied for detecting a misalignment between business processes and supporting software systems and identifying the process objects to be changed for restoring the alignment. A set of attributes, acting as indicators of a possible misalignment, and a strategy based on impact analysis, for detecting the impacts on the process components through the use of propagation rules from an initial set of changes, are proposed.

Section 2 describes the proposed alignment strategy. The subsequent section lists the requirements to be satisfied by a software environment to support the proposed alignment strategy. Final remarks are provided in the last section.

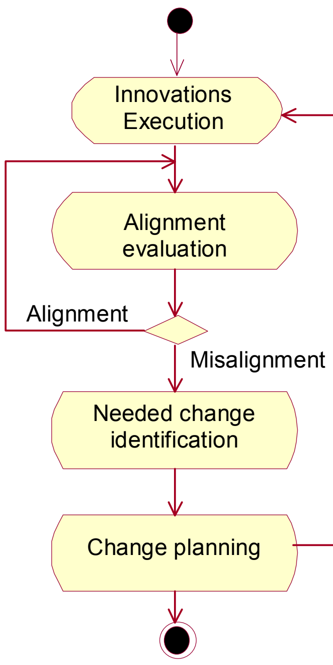

Fig. 1 UML activity diagram highlighting the activities of the proposed alignment strategy

2 Alignment strategy

Fig. 1 illustrates the proposed 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 of the constructed process models and the cooperation among different actors for their updating if evolution is needed. The formalism used in the current research is the Unified Modeling Language (UML) [11]; even if it can appear that it does not include all the necessary instruments and that it should be extended. Fig. 1 exploits this formalism. In particular, a UML activity diagram depicts the proposed alignment strategy. It is composed of the following four coarse grained phases:

Innovations 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 can consist of: a management 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: regarding the identification of the misalignment due to the performed change. The main problem faced in this task concerns how to measure and check the alignment. The solution adopted in this paper consists of the identification 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, 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 phase should be executed periodically, even when innovations have not been officially introduced but the change is 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 modification. In the current research, the identification of the process objects to be considered in the alignment actions is obtained by considering the dependence relations existing among the process objects, and propagating the change among them by applying impact analysis techniques.

Change plan: once identified the objects to be involved in the alignment actions, the innovation activities to be performed have to be identified. To fulfill this purpose, some approaches have been defined in literature based on measurement activities and critiquing techniques [2, 4, 5]. The identified actions could not be applied all in the same moment but on the basis of some priorities. Therefore, the phase aims at identifying the alignment actions and planning their execution modality.

This paper concerns mainly the second and third phases of the strategy above. In particular, the next two subsections outline the evaluation of the alignment and the impact analysis technique applied in the business context.

2.1 Detecting the misalignment

To be able to detect the misalignment, it is needed to codify what is the alignment and how to evaluate it. The way investigated here is to consider some appropriate parameters, evaluated by considering both business process and supporting software systems, and assuming values up to certain given threshold for indicating alignment. A first set of parameters can be:

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

Technological Adequacy for activity i (Ta_i), indicating how adequate is the support of a set of software system components for activity *i*. 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. This parameter can also be formulated in a more complex manner and consider other information besides those indicated.

The organization can represent the wished alignment by defining a set of thresholds, *Thre_TC*, $Thre\_Ta_i$. The considered business process and supporting software systems are aligned if:

$$Thre\_TC >= TC \text{ and/or } Thre\_Ta_i >= Ta_i, \text{ for all activity } i.$$

If a change causes $Thre\_TC < TC$, it means that a management (introduction and/or modification of an activity) or technological (modification of the software system) innovation decreases the technological coverage causing misalignment. Analogously, $Thre\_Ta_i < Ta_i$, for a certain *i*, indicates misalignment in activity *i*.

In any case, it can also happen that a change does not decrease the parameters values, preserving the alignment. This can happen if, for example, a useless activity is deleted or technological innovations increases both TC and Ta_i, for all *i*.

2.2 Propagating the change through Impact Analysis

Impact analysis is a technique very diffused 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* [10]. 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 vary for including not only source code but also any operational documentation and procedures.

In the business process management field, the impact analysis techniques can represent a useful instrument for identifying the effect of a change to the business process and supporting software systems. In this case, the objects to be considered do not include just the software system and its components but also the organization, business processes, process activities and so on, and the actions to be identified consist of changes to be performed on business process activities and software systems components in order to keep them aligned.

Assumed that the business process is modeled by using a given formalism, for example UML, it is necessary to understand the dependence relations among the objects to be analyzed As already stated, this research is focused on process components and supporting software systems. A first list of dependences is listed in the following:

- `include(process, activity)`, as a business process includes a set of activities;
- `depend_on(activity1, activity2)`, as a business process activity can depend on another activity, because, for example, the first activity processes information needed to the second one, that cannot start if the previous is not completed;
- `use(process, software_system)`, as a business process can be supported by one or more software systems;
- `composed_of(software_system, component)`, as a software system is composed of a set of software components;
- `depend_on(component1, component2)`, as a software component can depend on another software component in different way: use, data, inheritance, composition and so on.

The considered objects and dependence relations form a graph, called *Dependency Graph*.

Once the dependence relations have been identified, for performing Impact Analysis activities, it is necessary to define:

- the *type of modifications*, that describes the different way an object may be modified; in particular:
  - an activity may be modified in the way it is executed or in the interaction modality with the other activities;
  - a software system component may be modified in its interface or its implementation;
- the *propagation rule*, describing when and how to propagate a modifications to the connected objects. For example:
  - if an activity is modified in the way it is executed, the change can impact the software system components supporting it;
  - if an activity is modified in the interaction modality with the other activities, it can impact both the other activities from/on which it depends and the software system components supporting it;
  - if a software system component is modified in its interface, the change can impact the activities using the software component and the other software system components from/on which it depends;

- if a software system component is modified in the implementation, this change may impact the other software system components from/on which the considered software component depends.

A propagation rule may be formalized in the following way [7]: `(m1, c1, r, c2) --> m2`, which means: if a modification of type ml is applied to an object `01` of type `cl` which is connected to an object `02` of type `c2` through a link of type `r`, then a modification of type `m2` is to be applied to `02`.

The impacts on an object can be recursively propagated in order to obtain the complete set of impacts deriving from a modification.

The change requirements should be identified so that the alignment characteristics are kept for the aligned activities and restored for the misaligned ones. Guidelines for identifying the change requirements can be found in [6].

3 Tools for modeling, Impact Analysis and visualization

To effectively support the presented strategy a software environment should implement a set of requirements that supports the proposed strategy, by including adequate support for monitoring the alignment by measuring parameters and executing impact analysis. In particular, the main requirements of the software environment should be:

Req-1. It should be provided all the graphical support for designing the process model by using a standard description language;

Req-2. It should be provided an adequate support for designing the software system architecture;

Req-3. It should be included a support for measuring a set of chosen parameters and comparing them with given threshold

Req-4. Unsatisfactory parameter values have to be associated to consistent critiques, both in a textual and graphical environment;

Req-5. It should be included cyclic monitoring activities;

Req-6. It should be provided all the support for designing the relations between the business processes, activities and the software system as well as the software components;

Req-7. It should include supports for finding and visualizing the misalignment between business processes, activities and software systems.

Req-8. Supports for defining the propagation rules for impact analysis should be provided;

Req-9. It should support the execution of the propagation rules and the visualization of the impacts of changes;

Req-10. All the stored data have to be kept for future analysis and comparison.

The technologies to be used can consider previous experiences based on the exploitation of open source code. In particular, WebEv+ [2], Web for the Evaluation+, has been previously implemented, for supporting business process evolution through critiquing techniques, by integrating the WebEv environment [3] and the open source tool ArgoUML [1]. WebEv+ implements some of the listed requirements and can represent a component of the software system.

4 Conclusion

This position paper proposes 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 an alignment intervention can cause 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.

Further effort is required for better formalizing the strategy and introduced techniques. Moreover, their adequate usage requires the support of a software environment. As intermediate solutions already exist, the realization of the software environment can be made easier by exploiting them.

Before considering as operative a strategy and related tools, experimentation is needed. With this in mind, an operative reality has been identified. It concerns the Regional Centre of Competence in Information and Communication Technology, CRdC ICT, operating in Benevento, Italy. The Centre, involving many research and industrial partners, aims at analyzing, defining and realizing hardware and software platform to allow the provision of network services and implementation of advanced technologies.

References

1. ArgoUml. http://argouml.tigris.org/.
2. Aversano, L., Bodhuin, T., Canfora, G., Esposito, R., Tortorella, M.: Critiquing Business Processes for their Evolution towards E-Business. Proc. of ACM Symposium on Applied Computing, SAC 2004, Nicosia, Cipro, ACM Press, March 14-17 (2004), 1351-1358
3. Aversano, L., Bodhuin, T., Canfora, G. and Tortorella, M., WebEv – a Collaborative Environment for Supporting Measurement Frameworks. in Proc. of Hawai'i International Conference on System Sciences, HICSS-37 (Island of Hawaii, Big Island, January 2004), IEEE Comp.Soc. Press.
4. Aversano, L., Esposito, R., Mallardo, T., Tortorella M.: Supporting Decisions on the Adoption of Re-engineering Technologies. In proc. of CSMR 2004, 8th European Conference on Software Maintenance and Reengineering, Tampere, Finland, IEEE Comp.Soc. Press, March 24-26 (2004), 95-104

5. Aversano, L., Esposito, R., Mallardo, T., Tortorella, M.: Evolving Legacy System towards eBusiness. In: Managing Corporate Information Systems Evolution and Maintenance, Idea Group publishing: Hershey PA (2005), 295-325
6. Aversano, L, Tortorella, M.: An assessment strategy for identifying legacy system evolution requirements in eBusiness context. Int. Journal of Software Maintenance and Evolution: Research and Practice, Special Issue for CSMR 2003, John Wiley & Sons, (2004) to appear.
7. Barros, S., Bodhuin, T., Escudie, A., Queille, J.P., Voidrot, J.F.: Supporting Impact Analysis: a Semi-Automated Technique and Associated Tool. In Proc. of International Conference on Software Maintenance, Nice, France, IEEE Comp. Soc. Press, September 16-20 (1995), 42-51
8. Hammer, M., Champy, J.: Reengineering the Corporation: a Manifesto for Business Revolution, HarperCollins: New York NY, (1993)
9. Jacobson, I., Ericsson, M., Jacobson, A.: The Object Advantage: Business Process Reengineering with Object Technology, ACM Press: New York NY, (1995)
10. Queille, J.P., Voidrot, J.F., Wilde, N., Munro, M.: The Impact Analysis Task in Software Maintenance: A Model and a Case Study. Proc. of International Conference on Software Maintenance, Victoria, Canada, IEEE Comp.Soc. Press, September 19-23 (1994)
11. Rational Software, UML Notation Guide versione1.1, http://www.rational.com/uml/resources/documentation/media/ad970805_UML11_Notation2.pdf (1997)