Data & Knowledge Engineering xxx (xxxx) xxx–xxx

Contents lists available at ScienceDirect

Data & Knowledge Engineering

journal homepage: www.elsevier.com/locate/datak

Ensuring the canonicity of process models

Henrik Leopold[a,*], Fabian Pittke[b], Jan Mendling[b]

[a] Vrije Universiteit Amsterdam, De Boelelaan 1081, 1081 HV Amsterdam, The Netherlands
[b] WU Vienna, Welthandelsplatz 1, 1020 Wien, Austria

ARTICLE INFO

Keywords:
Conceptual modeling
Business process
Canonical representation
Non-canonicity patterns
Pattern refactoring

ABSTRACT

Process models play an important role for specifying requirements of business-related software. However, the usefulness of process models is highly dependent on their quality. Recognizing this, researches have proposed various techniques for the automated quality assurance of process models. A considerable shortcoming of these techniques is the assumption that each activity label consistently refers to a single stream of action. If, however, activities textually describe control flow related aspects such as decisions or conditions, the analysis results of these tools are distorted. Due to the ambiguity that is associated with this misuse of natural language, also humans struggle with drawing valid conclusions from such inconsistently specified activities. In this paper, we therefore introduce the notion of canonicity to prevent the mixing of natural language and modeling language. We identify and formalize non-canonical patterns, which we then use to define automated techniques for detecting and refactoring activities that do not comply with it. We evaluated these techniques by the help of four process model collections from industry, which confirmed the applicability and accuracy of these techniques.

1. Introduction

Process models are an important means to specify requirements in business-related software development projects [1]. Nevertheless, practitioners often struggle with the definition of fully correct and meaningful models [2]. The reasons for this are manifold. For instance, many modelers in practice have limited modeling experience [3], modeling projects often involve an overwhelming number of models [3], and the work of modelers involved in one project is difficult to coordinate [4]. The implications of incorrect and inconsistent models are severe. In the worst case, they entail wrong design decisions and a considerable increase of the overall development costs [5,6].

To ensure process model correctness and consistency, researchers proposed several automated analysis techniques. Such techniques can, for instance, check whether a process model contains deadlocks [7], is compliant with expected behavior [8], and meets predefined naming conventions [9]. The shortcoming of these techniques is, however, that they already make assumptions about the way modelers have used natural language to label the process model activities. As a result, these techniques are hardly of any help if the logic of modeling and routing elements is textually described in activity labels. As an example, consider the activity "*Consult expert and prepare report*" from one of the models we encountered in practice. Apparently, this activity label consists of two separate activities, i.e., "*consult expert*" and "*prepare report*", which are linked using the conjunction "*and*". The problem is that the execution semantics between these linked activity parts is not clearly defined. The word "*and*" might either refer to a parallel or a sequential execution. The specification of the activity as in this example mixes natural language and control structure in a way that is inherently ambiguous. This makes it impossible to draw valid conclusions from formal analysis results and, thus, difficult to develop

* Corresponding author.
E-mail addresses: h.leopold@vu.nl (H. Leopold), fabian.pittke@wu.ac.at (F. Pittke), jan.mendling@wu.ac.at (J. Mendling).

http://dx.doi.org/10.1016/j.datak.2017.03.010
Received 6 July 2016; Accepted 26 March 2017

process-related systems that are in line with the specification.

In this paper, we address this problem by introducing the notion of *canonicity* to prevent the mixing of natural language and modeling language. Based on this notion, we automatically check for problems caused by the violation of canonicity and point to reworks for resolving them. More specifically, we provide the following contributions. First, we introduce the notion of canonicity for process models and provide an operationalization of the concept. Second, we formalize a number of non-canonical patterns we discovered in models from practice. Third, we develop algorithms to recognize whether a given label suffers from these patterns and to refactor the detected cases into canonical model fragments. In order to demonstrate the applicability of the proposed techniques, we conduct extensive experiments with four real-world process model collections.

The rest of the paper is structured as follows. Section 2 illustrates the problems of non-canonical process model activity labels and reviews how prior research approaches have addressed this issue. Section 3 explains how we operationalize the notion of canonicity and our strategies to recognize and refactor instances that do not comply with it. Section 4 evaluates our techniques with process model collections from practice. In Section 5, we discuss implications and limitations of our work, before Section 6 concludes the paper.

2. Background

This section introduces the background of our research. First, Section 2.1 illustrates the problem of mixing modeling language and natural language and reflects upon the implications of non-canonical activities for system analysis and design. Section 2.2 discusses in how far prior research from the field of process model analysis has addressed the issue of non-canonical activities.

2.1. Problem illustration

In order to elaborate on the problem of non-canonical process model activities, we use the exemplary BPMN process model depicted in Fig. 1. It shows a process model of a company receiving new goods that are meant to be stocked in the warehouse. The process starts by screening the documents of the delivery. Then, the delivery is identified within the company before the inspection of the delivery begins. Afterwards, it needs to be determined if the delivery is complete or not. Depending on the result, the missing items are either requested from the supplier or the inventory is updated and the delivery documents are archived. Finally, the delivered goods are moved to the warehouse.

Fig. 1 shows several process model activities that are non-canonical. For example, the activity "*Screen delivery documents if necessary*" instructs the model reader to quickly check the delivery documents. Implicitly, it further instructs the reader to make a decision about the necessity. This means it would be more consistent to convey this decision through an exclusive choice gateway. The next example concerns the activity "*Delivery identification before inspection*". This activity describes a sequence of two activities with a particular order. Indeed, we first need to identify the delivery within the company and then inspect it accurately. This means it would be more consistent to create two separate activities in a sequence. Although both cases represent an inconsistent mix of

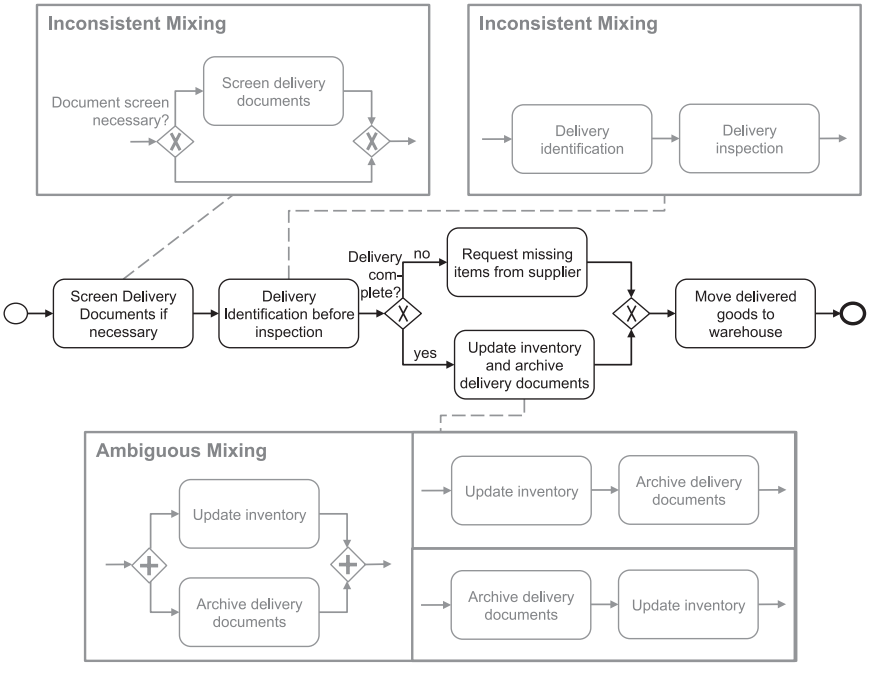

Fig. 1. Process Model with Mixing Natural Language and Modeling Language.