Information Systems 39 (2014) 108–133

Contents lists available at ScienceDirect

Information Systems

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

Configuration vs. adaptation for business process variant maintenance: An empirical study

Markus Döhring [a],*, Hajo A. Reijers [b], Sergey Smirnov [c]

[a] *SAP AG, Dietmar-Hopp-Allee 16, 69190 Walldorf, Germany*
[b] *Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands*
[c] *SAP AG, Am Schimmersfeld 5, 40880 Ratingen, Germany*

A R T I C L E I N F O

Article history:
Received 27 August 2012
Received in revised form
11 April 2013
Accepted 3 June 2013
Recommended by: M. Weske
Available online 9 July 2013

Keywords:
Process modeling
Process variants
Process configuration
Process adaptation
User experiment
Model maintainability

A B S T R A C T

Many approaches for process variant management employ a reference model for deriving a target variant either using configuration or adaptation mechanisms. What is missing at this stage is empirical insight into their relative strengths and weaknesses. Our paper addresses this gap. We selected C-YAWL and vBPMN for a comparative, empirical user study. Both approaches center on a reference process, but provide different types of configuration and adaptation mechanisms as well as modularization support. Along with this aspect, we investigate the effect of model complexity and professional level on human process variant modeling performance. Given unlimited processing time, we could not show that complexity or the participant's professional level significantly impacts the task success rate or user contentment. Yet, an effect of model complexity can be noted on the execution speed for typical variant maintenance tasks like the insertion and deletion of process steps. For each of the performance measures of success rate, user contentment and execution speed, vBPMN performs significantly better than C-YAWL. We argue that this is due to vBPMN's advanced modularization support in terms of pattern-based process adaptations to construct process variants. These insights are valuable for advancing existing modeling approaches and selecting between them.

1. Introduction

The ability to rapidly tailor a process to changing business requirements is among the top drivers of companies to employ Business Process Management (BPM) technology [1,2, p. 3]. In this context, it is often the case that new business requirements have to be taken into account over time. Such requirements may supplement existing ones, leading to the need for a slightly different behavior of the business process as executed previously. This motivates an efficient *process variant* modeling approach. For example, a sales order process that is run by a global company enforces the execution of a liquidity check at the start of the process for customers in Asia only, while for European customers this step is skipped. At some point in time, the company may want to execute a liquidity check for distinct European countries, but only at the end of the process. This necessitates the introduction of an additional process variant.

According to [3][4, p. 4], variants of a process model are defined as “similar-but-different” from each other, i.e. they have at least one feature in common and one feature in which they differ. In practice, however, this definition of the term *variants* tends to be problematic, since one often finds at least one commonality or invariant [5] between two objects. It seems more practical to adopt a definition as from [5], where it is clearly stated that the delta between two objects should be

* Corresponding author. Tel.: +49 1711552120.
E-mail address: markus.doehring@sap.com (M. Döhring).

0306-4379/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.is.2013.06.002

small compared to their commonalities. For this paper, we consider structural characteristics like sequencing or branching behavior in the process graph as relevant features [6].

A typical pitfall that should be avoided when modeling process variants notably relates to the creation of redundancy by applying a copy-and-paste (multi-model) approach [7]. In such an approach, an existing process model is cloned and tailored to the new business requirement. Since there is no shared part list for the (loosely) corresponding process models, it is very hard to enforce global changes if the number of variants is high. One example would be the insertion of a new task that should be executed within all process variants, requiring the manual tailoring of each single process model.

Furthermore, as business requirements change over time, maintenance operations on the distinct process variants need to take place. In this respect, maintenance operations may, for example, relate to the variant-specific insertion, skipping or re-routing of process steps [8]. Two important characteristics which determine model *maintainability* are *understandability* and *modifiability*. This means that to properly maintain a process model, a user is not only required to correctly understand the process model, but must also be enabled to properly modify it according to a specified adjustment task. For this paper we subsume the terms *understandability* and *modifiability* under *maintainability*. The reason is that a lot of work related to software measurement exists, which considers understandability as an influencing factor for maintainability [9–13].

In order to address the challenges we described above, a broad variety of modeling approaches for process variants has emerged in recent years, e.g. [7,14–23]. For these approaches, a comprehensive survey which classifies them according to multiple feature dimensions was conducted in [24]. While we do not claim these feature dimensions to be exhaustive, five of the most commonly addressed dimensions in scientific literature are explained in the following:

Variant construction direction. Process variants can be constructed and maintained using generally different strategies [4,7,18], including process *configuration* and *adaptation*.

When using process *configuration*, typically the first step is to create a reference process which comprises the behavior of all considered variants. From this all-embracing reference process model, a variant can be derived by eliminating elements which are not relevant for the given context. This corresponds to one way of *process model abstraction* [25].

When using process *adaptation*, the reference process is not necessarily constructed as the superset of all variants. Instead, a set of change operations [8] as, for example, the insertion, deletion, conditional-skipping or loop-embedding of tasks are defined. An appropriate reference model can then be selected by minimizing the change operations, potentially considering variant usage frequency, which need to be applied to the reference process for obtaining the required process variant [26]. Since these change operations can be perceived as extensions to the reference process, this strategy is strongly related to the concept of *inheritance* in process models [27].

Modularization support. Modularity is usually referred to as a system property, which states that the system is composed of smaller subsystems. These subsystems in turn are independently manageable and function together as a whole [28,29]. Decomposition is referred to as a stricter subconcept of modularity, in which modules need to be designed such that the interdependencies to other modules are minimized. Conveying this to process variants, modularization manifests itself primarily in variable regions spanning multiple process elements, which can be subject to change as a whole. Modularization may also manifest itself in reusable process fragments or respectively change macros which apply complex modifications on the reference process, as for example the wrapping of a process fragment into a loop construct or timeout exception handler.

Runtime variant construction. In some cases it is necessary to alter an instantiated process during runtime to a variant which had not been considered before starting it [8]. The ability to construct new variants at runtime, for example, refers to the inserting a new task, such as a special approval task.

Data-flow and resource variability. Besides the modeling of control-flow variants, variability in processes can also relate to many other perspectives [30]. Most prominent are the data-flow and the resource perspective [31, p. 2]. Data-flow variants for example specify different types of objects to be passed within a static control-flow, while resource assignment variants specify different processors like clerks or computers for a process task.

Existing approaches for process variant management concentrate on different parts of the above dimensions. Most of them feed their claim to support the maintainability of process variant models by presenting case studies, which provide an impression on how a process variant model would be realized using the respective approach. No quantitative empirical evidence, however, exists on the actual benefits and drawbacks of the distinct approaches for the human process variant modeler, for instance regarding the maintainability of the process model. Moreover, only few insights are reported regarding the scalability of the approaches, i.e. what happens if the variant model gets complex and needs to be maintained over time.

Since control-flow is considered as the essential perspective in process modeling [31, p. 2] and very limited empirical prior work exists on process variant modeling, in this work we first focus on the two control-flow related feature dimensions described before, i.e. variant construction direction and modularization support. We leave the examination of other feature dimensions to future work. For the variant construction direction, effects on maintainability have not yet been thoroughly examined. This dimension is, however, recognized as the main classification criterion for approaches to tailor reference models [4]. For modularization support, a general positive effect on understandability has already been established empirically in [28]; moreover, modularization is described as a subcharacteristic of maintainability in the ISO 25010 standard [32].