Information Systems 39 (2014) 45–79

Contents lists available at ScienceDirect

Information Systems

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

# Automated runtime repair of business processes

N.R.T.P. van Beest [a,*,1], E. Kaldeli [b,1], P. Bulanov [b], J.C. Wortmann [a], A. Lazovik [b]

[a] Department of Operations, Faculty of Economics and Business, University of Groningen, Nettelbosje 2, 9747 AE Groningen, The Netherlands

[b] Distributed Systems Group, Johann Bernoulli Institute, University of Groningen, Nijenborgh 9, 9747 AG Groningen, The Netherlands

ARTICLE INFO

*Article history:*
Received 17 September 2012
Received in revised form
8 July 2013
Accepted 10 July 2013
Recommended by: L. Wong
Available online 18 July 2013

*Keywords:*
Process interference
Business process management
Run-time process repair
e-Government

ABSTRACT

Concurrent business processes frequently suffer from mutual interference, especially in highly distributed service environments, where resources are shared among different stakeholders. Interference may be caused by supposedly stable case-related data, which are modified externally during process execution and may result in undesirable business outcomes. One way to address this problem is through the specification of dependency scopes, that cover critical parts of the process, and intervention processes, which are triggered at runtime to repair the inconsistencies. However, for complex processes, the manual specification of the appropriate intervention processes at design time can be particularly time-consuming and error-prone, while it is difficult to ensure that all important intervention cases are taken into account. To overcome this limitation, we propose an approach for automating the generation of intervention processes at runtime, by using domain-independent AI planning techniques. This way, intervention processes are composed on the fly, taking into account the characteristics of the business process in execution, the available compensation activities, and the properties that have to be fulfilled to recover from the erroneous situation. A prototype has been implemented and evaluated on a real case study of a business process from the Dutch e-Government.



## 1. Introduction

Modern organizations are moving from traditional, proprietary and locally managed Business Process Management Systems (BPMS) to BPMS where more and more tasks are outsourced to third party providers and resources are shared among different stakeholders [31,18,10]. The application of principles inspired by Service Oriented Architectures (SOA) enables the integration of interoperable, local or remote services within a business process (BP), aiming at adaptability and reuse. In such an open and dynamic setting, BPs can no longer be considered in isolation, since data resources are not necessarily proprietary to the organization, but are simultaneously shared with other external actors and processes. e-Government is a typical area that is characterized by multiple concurrently executing knowledge-intensive processes, which access and modify commonly shared resources such as citizen data, information reported by external contracted parties, etc. In such a context, traditional verification techniques for workflows and data-flows, e.g. [53] are not sufficient for ensuring the correctness of such BPs, since they assume that process and data interactions are predefined in advance. However, not all interactions are known or prespecified, since data can be simultaneously accessed and modified by different processes, with no obvious relation to the BP in progress. Disregarding the interdependencies with external actors

* Corresponding author. Tel.: +31 503 638 562.
*E-mail address:* n.r.t.p.van.beest@rug.nl (N.R.T.P. van Beest).

[1] These authors have contributed equally to this paper.

0306-4379/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.is.2013.07.003

and other processes may lead to inconsistent situations, which potentially result in undesirable business outcomes. The situation where undesirable business outcomes are caused by data modifications of some other concurrently executing process is known as *process interference* [63,55].

Process interference occurs far more often than most people realize. Processes are developed under the assumption that case-related data are stable, and this assumption is in general not true. As soon as case-related data are changed, processes may yield wrong results. In many cases, however, these wrong results do not lead to immediate software errors. Consequently, there exists the incorrect impression that the process runs well. Nevertheless, in the real world these interference situations lead to erroneous situations. These may refer to proceeding to activities based on obsolete information (e.g. delivering a product to some address that is not valid anymore), the repetition of activities that have already been fulfilled by some other process (e.g. multiple orders or invoices), or disregarding events that call for compensation activities or the process to hold (e.g. some necessary condition ceases to hold). These errors in the real world lead to customer complaints, legal cases, and many untraceable societal costs [56]. However, their root cause, process interference, is overlooked in process management software architectures.

Most work about resolving process interference refers to failing processes or concerns design-time solutions [64,54]. In [55], a run-time mechanism is proposed, which uses dependency scopes and intervention processes to manage interference discovered during execution. *Dependency scopes* (*DSs*) are used to specify critical parts of the BPs whose correct execution relies on the accuracy of a *volatile* process variable, i.e. a process variable that can be changed externally during the execution of the process. If a volatile variable is externally modified while the execution flow resides within the range of the respective DS, a predefined *intervention process* (*IP*) is triggered as a response, with the purpose of resolving the potential inconsistencies stemming from this change event. In [57], the initial idea is enhanced with an algorithm, which automates the task of identifying the critical sections of a BP.

By using DSs, it is no longer necessary in the design of the process to incorporate checks prior to each activity whether an important data element has changed. As a result, the process designer does not need to know all potential process interactions in advance. However, a significant effort is required for manual specification of the IPs, since the appropriate IPs may differ considerably depending on the current execution state at which modification of a volatile variable occurred. For complex processes with numerous activities, it is very difficult and time-consuming to define IPs at design-time, as the amount of potential IPs may be particularly high. In addition to that, it is difficult to ensure that *all* important intervention cases are taken into account. Moreover, as the same BP may be deployed and used by more than one organization, different IPs have to be specified for each potential interference case at each organization.

The workload due to extensive manual configuration can be significantly reduced by automating the task of IP specification. Building upon the initial ideas presented in [55,57], in the current paper we propose the use of *domain-independent AI planning* to automate the process of specifying IPs. From this perspective, IPs are viewed as plans, which can be synthesized dynamically on the fly, by combining activities from the BP and available compensation operations. This composition takes place based on how the BP's knowledge about the world evolves during execution, and how this knowledge affects workflow tasks. In such a way, the manual work required by the domain designer is reduced to the specification of the dependency scopes and a high-level goal, which describes in a declarative way the desired consistent state that has to be reached in case of interference. To realize such a level of automation, the BP specification has to be enriched with appropriate semantic annotations, in terms of preconditions and effects. The restrictions imposed by the specific control flow of each BP are inferred automatically, by parsing the syntactic BP specification. An AI planner can then be employed to resolve inconsistencies at runtime in an automatic way, by resorting to declarative goals rather than predefined ad hoc processes.

By examining a realistic case-study from the Dutch e-Government, we show how the generation of IPs can be realized by state-of-the-art planning techniques. A fully-working prototype has been implemented, using the RuG domain-independent planner [26,28,29] based on dynamic Constraint Satisfaction Problem (CSP) techniques. The RuG planner is equipped with a number of features that go beyond classical planning, and are of particular relevance to the requirements associated with BP modeling and repair, such as the efficient handling of numeric variables, explicit support for incomplete knowledge, and a variety of effects beyond mere assignments. The framework has been implemented and evaluated on a real case-study from e-Government, in order to show the feasibility of the approach. The focus of the work presented herein is to address process recovery from inconsistencies that result from process interference. However, the overall approach of using domain-independent AI planning for BP reconfiguration is more general, and can be used to react to any kind of events.

The remainder of this paper is organized as follows. Section 2 provides an overview of related work, and Section 3 describes a possible interference scenario on a real case-study taken from Dutch e-Government, which plays the role of our running example. The architecture of the proposed framework is described in Section 4. In Sections 5 and 6, the definitions and methodologies concerning the proposed approach are presented. The implementation of the framework is described in Section 7. The performance of the implemented framework is evaluated in Section 8, and the overall conclusions are drawn in Section 9.

## 2. Related work

The work presented herein shares many aspects with different subfields of BP management, including work in the areas of BP recovery, adaptation and process interference. Various viewpoints on these issues are presented and compared in