The Journal of Systems and Software 113 (2016) 59–75

Contents lists available at ScienceDirect

The Journal of Systems and Software

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

Run-based exception prediction for workflows

Yain-Whar Si [a,*], Kin-Kuan Hoi [a], Robert P. Biuk-Aghai [a], Simon Fong [a], Defu Zhang [b]

[a] Department of Computer and Information Science, University of Macau, Macao
[b] Department of Computer Science, Xiamen University, China

ARTICLE INFO

Article history:
Received 31 January 2015
Revised 10 November 2015
Accepted 14 November 2015
Available online 26 November 2015

Keywords:
Workflow
Run
Temporal exception prediction

ABSTRACT

Events such as iteration of activities or lack of available resources can cause temporal exceptions in business processes. Exception prediction can improve the quality of workflow execution since preventive actions can be taken to reduce the occurrence of exceptions. Thus, it is crucial to provide an accurate and efficient temporal exception prediction capability for workflow management systems. In this paper, we propose a run-based exception prediction algorithm to predict temporal exceptions in workflows. The proposed algorithm is divided into two phases, design-time and run time. At design-time, all possible runs are generated from a workflow and their estimated execution time and mapping probability are calculated. At run time, temporal exceptions are predicted by analyzing the runs. Simulation experiments are performed to evaluate the proposed approach using five workflow models having different characteristics. Simulation experiments show that our approach is efficient and produces good results in prediction accuracy.

1. Introduction

A workflow is a collection of tasks organized to achieve business goals (Li and Yang 2005; Pan, Tang et al. 2005; Son, Kim et al. 2005; Dumas, Garcıa-Banuelos et al. 2011). Various aspects of workflow management systems have been extensively reported in the literature, such as verifying and optimizing workflows (Cao, Jin et al. 2013). One of the most important issues in workflow management is the time management for executing its activities. For instance, a workflow can be designed with a set of tasks to be executed by staff for delivery of goods. The process begins when an order for goods is received and ends when the goods are delivered to the customer and the payment is received. Any delay in completing the process can lead to financial loss, damaged reputation, or customer dissatisfaction. Therefore, predicting temporal exceptions and taking necessary measures to avoid delays in execution of the processes can significantly reduce costs and in the long run make businesses more competitive.

A Workflow Management System (WFMS) is designed to manage the workflows and organize the flow of tasks such that a task is performed by the right person/application (resource) at the right time (del Foyo and Silva 2008). Since time plays an important role in a WFMS, it is necessary to provide time-related constraints, such as bounded execution durations of tasks and absolute deadlines of workflows (Eder and Pichler 2005). However, some unexpected events can cause time violations as some of the tasks need to be repeated or necessary resources are unavailable. A temporal exception occurs when a time constraint defined at design time is violated during run time (Xie, Yu et al. 2009).

Exception prediction can be useful in improving the quality of workflow execution since preventive actions can be taken to reduce the occurrence of exceptions. When a workflow instance is predicted to have a high probability of violating its deadline constraints, a workflow administrator can allocate more resources, assigning a higher priority to reduce the likelihood of deadline violation (Grigori, Casati et al. 2001). Thus, it is crucial to provide an accurate and efficient temporal exception prediction capability in a WFMS.

A number of exception prediction methods have been proposed in the literature (Eder, Gruber et al. 2000; Eder and Pichler 2002; Leong, Si et al. 2012; Yu, Xie et al. 2013). However, these approaches do not consider a number of critical issues in the control flow of workflows. In this paper, we propose a novel temporal exception prediction method for workflows addressing these control flow aspects. Specifically, in our approach, the concept of runs (the complete execution path) of a workflow is used for predicting temporal exceptions.

1.1. Motivation

Various approaches to workflow exception prediction can be found in the literature (Eder, Gruber et al. 2000; Eder and Pichler 2002; Leong, Si et al. 2012; Yu, Xie et al. 2013). However, they

* Corresponding author. Tel.: +853 8822 4454; fax: +853 8822 2426.
E-mail addresses: fstasp@umac.mo (Y.-W. Si), neti1723@hotmail.com (K.-K. Hoi), robertb@umac.mo (R.P. Biuk-Aghai), ccfong@umac.mo (S. Fong), dfzhang@xmu.edu.cn (D. Zhang).

http://dx.doi.org/10.1016/j.jss.2015.11.024
0164-1212/

provide limited support for predicting temporal exceptions. Firstly recent work (Eder, Gruber et al. 2000; Eder and Pichler 2002; Yu, Xie et al. 2013) conducted exception prediction at a specific time point such as at the beginning or end of a task. Secondly existing work (Eder, Gruber et al. 2000; Eder and Pichler 2002; Leong, Si et al. 2012; Yu, Xie et al. 2013) ignores some situations in parallel control structures. Specifically, during prediction these algorithms choose a particular task of one of the branches of the AND gateway and make conclusions based on the result of that prediction. Therefore, simultaneous predictions at several locations on different branches can produce inconsistent conclusions. In our approach, a single unified conclusion for prediction at a specific time point is drawn based on the exception probability of the given workflow instance while taking into account all AND/XOR branches. Thirdly these algorithms only provide limited support for iterative control structures which is used to control the repetitive execution of one or more tasks. In this paper, we introduce a run-based prediction algorithm to overcome these problems.

1.2. Contributions

The main contributions of our run-based exception prediction algorithm are as follows:

(1) The proposed algorithm can predict exceptions in workflows involving parallel control structures while taking into account all parallel branches.
(2) It can not only predict exceptions at any time point but also can predict exceptions for workflows that consist of crossing loops and nested loops.
(3) Our algorithm can be used with additional parameters in exception prediction, for instance, the number of iterations for loops and the threshold for exceptions according to users' preferences.
(4) The proposed algorithm enables finer control of prediction processes since users are allowed to set thresholds in making predictions. Specifically, the proposed run-based exception prediction algorithm not only considers the choice probabilities in selective control structures, but also the exception probability of an affected-run, and exception probability of an instance.

2. Literature review

In (Eder, Gruber et al. 2000), researchers proposed an algorithm for handling alternative execution paths by augmenting each task node with temporal information. In (Eder, Gruber et al. 2000), two explicit time constraints are defined; they are lower (lbc) and upper (ubc) bound constraint, which are used to define the minimum and maximum time distance between source event and destination event. Based on these two time constraints, the algorithm computes the earliest point in time a task node can finish when the shortest (longest) path from start node to the task node is taken, called best case E-value (worst case E-value); and they compute the latest point in time a task node has to finish in order to meet the overall deadline when the shortest (longest) path is taken, called best case L-value (worst case L-value). The exception prediction is conducted by comparing the lbc and ubc to E-value and L-value to check whether the time constraints are violated or not. When a workflow is executing, at a given time point that a task ends its execution, if the current time is smaller than or equal to the best-case E-value of the task, it will predict no exception happened, otherwise exception happened is predicted; if the current time is greater than or equal to best case L-value of the task, it will predict an exception happened, otherwise no exception happened is predicted.

In (Eder and Pichler 2002), the researchers introduced an approach for enhancing the estimation of duration of workflow and the likelihood of time constraints violations by constructing a new structure called duration histogram, which is used to present temporal information and different probability values for different branches at XOR-split nodes. A duration histogram of a node is a (n x 2)-matrix, where each row holds one tuple (p, d), representing the probability p of the remaining execution time within d between this node and the end node. In this paper, we improve the workflow graph by attaching the duration histogram to every node; the exception prediction is conducted by checking the remaining execution time in the duration histogram for the current task node.

In (Leong, Si et al. 2012), the researchers introduce a critical path based approach for predicting the occurrence of temporal exceptions in workflows. In this approach, firstly it generates all possible execution paths at design time, and secondly it calculates the head deadline (HD) from the longest path and the earliest completion time (CET) from the shortest path according to the execution status during run time, and finally it compares CET to HD to make the prediction: if the CET is greater than HD, it will predict an exception happened, otherwise no exception happened.

In (Yu, Xie et al. 2013), the researchers propose an algorithm based on historical temporal data for handling workflow time exceptions (deadline constraint violations). In this algorithm, they introduce a set of time cumulative distribution functions, such as execution time, queuing time, and total processing time etc. In their approach, a time probability model of the process is used to model the time indeterminacy of the execution time of tasks. By analyzing the time probability model and the workload of the available resources, the algorithm predicts the temporal exceptions of the workflow.

The related work stated above have some limitations in exception prediction. First, exceptions cannot be predicted at any time point (Eder, Gruber et al. 2000; Eder and Pichler 2002; Yu, Xie et al. 2013). For instance, prediction must be performed at the beginning or end of a task. Second, no crossing and nested loops are supported (Eder, Gruber et al. 2000; Eder and Pichler 2002; Yu, Xie et al. 2013). Although it is supported in (Leong, Si et al. 2012), it has limitations in flexibility, as it predefines a maximum iterative number for each loop. In our approach, we not only consider crossing and nested loops, but also allow users to define the probability of iteration. Third, (Eder, Gruber et al. 2000; Eder and Pichler 2002; Leong, Si et al. 2012; Yu, Xie et al. 2013) use temporal information of a chosen node to make the exception prediction. Since they only focus on one specific node, simultaneous prediction on different nodes can cause inconsistent conclusions. For instance, simultaneous prediction at different nodes in AND branches can produce inconclusive results. Therefore, in this paper, we present an algorithm that takes into account all possible execution paths of the workflow instance for making a unified conclusion. Fourth, (Eder, Gruber et al. 2000; Leong, Si et al. 2012) provide a direct yes/no answer for exception prediction without considering probabilities. Therefore, users have no information about the likelihood of the occurrence of exceptions. Although (Eder and Pichler 2002; Yu, Xie et al. 2013) provide an approach which uses probabilities for exception prediction, these values are defined at design time and they are not updated during run time. Note that some execution paths defined at design time may not map into any of the execution paths during actual execution. In these situations, the accuracy of the prediction could be affected. In this paper, we propose an algorithm which calculates the exception probability of an instance at run time. Table 1 shows the comparison of the above four papers to our run-based exception prediction algorithm, where S, PS, and NS denote "Supported", "Partially-Supported", and "Not-Supported".

3. Background

In this section we define the fundamental elements of the workflow used in the rest of this paper.