Available online at www.sciencedirect.com
ScienceDirect
Data & Knowledge Engineering 61 (2007) 59–75
DATA & KNOWLEDGE ENGINEERING
www.elsevier.com/locate/datak

Scheduling-free resource management

Kees van Hee, Alexander Serebrenik, Natalia Sidorova *, Marc Voorhoeve, Jan van der Wal

Eindhoven University of Technology, Department of Mathematics and Computer Science, P.O. Box 513, 5600 MB Eindhoven, The Netherlands

Received 11 April 2006; received in revised form 11 April 2006; accepted 11 April 2006
Available online 24 May 2006

Abstract

We investigate a resource management policy that allocates resources based on the number of available resources only. We formulate a condition on resource requesting processes, called *solidity* that guarantees successful termination. Processes that do not satisfy this condition can be modified to become solid. We investigate performance of the resource management policy proposed by comparing it to the theoretically found optimum for the special case. Our method can be applied in, for example, project management systems, orchestration software and workflow engines.

Keywords: Business process; Workflow; Resource management; Correctness; Scheduling

1. Introduction

Workflow nets [1–4], a special class of Petri nets, are frequently used to model business processes. In business processes, three elements are essential: *cases* to be processed, *tasks* to be performed on the cases and *resources* needed to perform these tasks. The execution of a specific task for a specific case is called *activity*. Traditionally, models of workflow nets emphasize the partial ordering of activities in the process and abstract from any resources needed, such as money, machinery or manpower. Petri nets are well-suited for modeling resource dependence of activities [6,11,14,15,22], so it is natural to extend workflow nets with resources. The so-called resource-constrained workflow nets [18] assume *durable* rather than *consumable* resources. The addition of resource dependencies may introduce deadlocks in an otherwise well-designed (*sound* [1]) workflow net.

* Corresponding author. Tel.: +31 40 2473705; fax: +31 40 2463992.
E-mail addresses: k.m.v.hee@tue.nl (K. van Hee), a.serebrenik@tue.nl (A. Serebrenik), n.sidorova@tue.nl (N. Sidorova), m.voorhoeve@tue.nl (M. Voorhoeve), jan.v.d.wal@tue.nl (J. van der Wal).

0169-023X/$ - see front matter
doi:10.1016/j.datak.2006.04.008

Assessment of business process models involves *correctness* and *efficiency*. Correctness requirements include *proper termination*: the fact that all open cases can be completed and claimed resources returned from any reachable state, provided the initial number of resources exceeds some fixed minimum. Efficiency criteria can be divided into *quality of service* (QoS) and *costs of operation* (CoO) criteria.

We consider business process models with resources belonging to a single class. Cases are independent, i.e. they only influence each other via the resources they need. They communicate with a resource manager by claiming and releasing resources in various quantities. The resource manager (human or software) decides when to grant the claims. No preemption or priority requests are allowed. The resource manager bases its decisions on the *global state* of the process, i.e. the number of cases, the state of each case and the number of available resources.

The first task of resource management is to ensure that needed resources are always eventually available, which involves *dynamic* scheduling. In contrast to *static* scheduling problems, in which all cases to be handled are known, in *dynamic* scheduling problems the arrival, routing and resource consumption of cases cannot be predicted. The banker's algorithm of Dijkstra [12] ensures correctness in this way. This algorithm considers the number of available resources and for each case the maximal number of resources needed by the case (credit limit) and the number of resources claimed but not yet returned so far (debt). Dijkstra's algorithm considers correctness only, but it leaves room for prioritizing between cases that need resources when they become available. These priorities can be based on heuristics like FCFS (first come first served), SPT (shortest remaining processing time) or EDD (earliest due date) and they can be compared for efficiency.

In [18], we considered processes with the *scheduling-free resource management policy*, i.e. the policy when a resource request may be granted whenever enough resources are available to satisfy the request. We also formulated a property regarding the resource behavior of cases, which we call *solidity*. A business process consisting of any number of cases and working under the scheduling-free resource management policy is guaranteed to terminate properly if and only if all cases are solid. Moreover, the process is robust w.r.t. the addition of cases during the process run, i.e. it stays proper terminating.

Cases that are not solid can be *solidified* by setting *thresholds*: additional resources that need to be available for each resource claim. Resources are granted to a task only if the number of available resources exceeds the number of claimed resources plus the threshold. Thresholds are chosen according to the following policy: *before committing resources to a case, make sure that there are enough resources available to allow its completion independent of other cases*. The thresholds can be determined in advance, when the business process is defined. Setting thresholds is akin to a well known approach in production control [9].

In this paper, which is an extension of [19], we investigate resource scheduling based on solidification. To determine thresholds that perform well, an iterative, simulation-based approach is proposed. We illustrate our approach with a small example inspired by the construction industry. Resource management based on thresholds has a runtime computational complexity that does not depend on the number of active cases, which makes it very suitable for workflow management systems. It is interesting to compare the performance of the robust threshold-based resource managers to more sophisticated ones, thus investigating the price of coordination (cost of robustness). For a tiny (tandem queue) example, we did compute an optimal global scheduler by means of Markov decision theory [21] and compared the performance of the robust and optimal resource schedulers.

Related work The problem of scheduling shared resources in flexible manufacturing systems has been studied extensively, specifically by modelling them as Petri nets (see [11,14,15,22,23] for an overview of works in this field). Like in our work, most of these works use invariants to give a necessary condition for the correctness of the use of resources. However, the authors focus on extending a model that represents the production process with a scheduler in order to avoid deadlocks and to use resources in the most efficient way; e.g. [14,15] propose a control policy for deadlock prevention based on ensuring that no siphon can be emptied. As mentioned above, our goal is to build a scheduling-free resource manager.

In [6] the authors consider structural analysis of Workflow nets with shared resources. Unlike this work we consider systems where the number of available resources can vary. So we require that the system should work correctly for any number of cases and resources.

The works [10,13] focus on the verification of LTL\X properties in networks of processes that interact by sharing resources. Both [10,13] use cut-offs to reduce their verification problems to finite-state model checking.