Information Systems 36 (2011) 1026–1043

Contents lists available at ScienceDirect

Information Systems

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

Soundness verification for conceptual workflow nets with data: Early detection of errors with the most precision possible

Natalia Sidorova *, Christian Stahl, Nikola Trčka

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

ARTICLE INFO

Available online 5 May 2011

Keywords:
Conceptual workflow models
Soundness
Refinement
Verification
Correctness
May/must semantics

ABSTRACT

A conceptual workflow model specifies the control flow of a workflow together with abstract data information. This model is later on refined by adding specific data information, resulting in an executable workflow which is then run on an information system. It is desirable that correctness properties of the conceptual workflow are transferable to its refinements. In this paper, we present *classical workflow nets extended with data operations* as a conceptual workflow model. For these nets, we develop a novel technique to verify *soundness*. An executable workflow is sound if from every reachable state it is always possible to terminate properly. Our technique allows us to analyze a conceptual workflow and to conclude whether there exists at least one sound refinement of it, and whether any refinement of a conceptual workflow model is sound. The positive answer to the first question in combination with the negative answer to the second question means that sound and unsound refinements for the conceptual workflow in question are possible.

1. Introduction

Information systems are a key technology in today's organizations. Prominent examples of information systems are enterprise resource planning (ERP) systems and workflow management systems. Processes are the core of most information systems [1]. They orchestrate people, information, and technology to deliver products. In this paper, we focus on *workflows*. A workflow refers to the automation of a processes by an IT infrastructure, in whole or in part [2].

A workflow is usually iteratively designed following a bottom-up approach. First, the control flow of the workflow is modeled. The control flow consists of a set of coordinated tasks describing the behavior of the workflow. Later, the control flow is extended with abstract data information. The resulting model is an abstract or *conceptual workflow model*; it is typically constructed by a business analyst. This conceptual model can be used for purposes of documentation, communication, and analysis. It may abstract from concrete data values, such as the condition of an if-then-else construct, and it usually does not specify how concrete data values are stored.

To actually execute this workflow on a workflow management system, the conceptual workflow model is instantiated with full detail—a task typically performed by business programmers (who often have insufficient background knowledge of the process) and not by the business analysts themselves. For example, the business programmer specifies concrete data values and how they are stored. The modeling of abstract and executable workflows is supported by industrial workflow modeling languages available on the market.

Designing a workflow model is a complicated and error-prone task even for experienced process designers. For a fast and, therefore, cost-efficient design process, it is important that errors in the model are detected during

* Corresponding author. Tel.: +31 40 247 3705; fax: +31 40 246 3992.
E-mail addresses: n.sidorova@gmail.com, n.sidorova@tue.nl (N. Sidorova), C.Stahl@tue.nl (C. Stahl), N.Trcka@tue.nl (N. Trčka).

0306-4379/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.is.2011.04.004

the design phase rather than at runtime. Hence, verification needs to be applied at an early stage—that is, already on the level of the conceptual workflow model.

Formal verification of a conceptual workflow model imposes two challenges. First, it requires an adequate formal model. With adequate we mean that the model captures the appropriate level of abstraction and enables efficient analysis. So the question is, *how can we formalize commonly used conceptual workflow models?* Second, verification must not be restricted to the control flow, but should also incorporate available data information. Thereby the conceptual workflow model may specify abstract data values that will be *refined* later on. Here the question is, *can we verify conceptual workflows in such a way that the results hold for any possible data refinement* (i.e., for all possible executable versions)?

In this paper, we investigate these two questions. We focus on one of the most established requirement for workflow correctness, the *soundness property* [3]. Soundness guarantees that the workflow has always the possibility to *terminate* and every task of the workflow is *coverable* (i.e., can potentially be executed). Termination ensures that the workflow can during its execution neither get stuck (i.e., it is deadlock free) nor enter a loop that cannot be left (i.e., the workflow is livelock free), whereas coverable excludes dead tasks in the workflow. However, current techniques for verifying soundness are mostly restricted to the control flow and do not consider data information.

To answer the first question, we propose *workflow nets with data* (WFD-nets) as an adequate formalism for modeling conceptual workflows. A WFD-net is a workflow net (i.e., a Petri net tailored toward the modeling of the control flow of workflows) extended with conceptual read/write/delete data operations and guards. A guard defines an additional constraint that must be fulfilled to enable a transition (i.e., a task). WFD-nets generalize our previous model from [4] by supporting arbitrary guards—previously, only predicate-negation and conjunctions were allowed. The syntax of WFD-nets is actually similar to the syntax of the Protos tool of Pallas Athenas, which is the leading process modeling tool in the Netherlands.

Example 1. We illustrate the idea of modeling conceptual workflows with WFD-nets with the WFD-net in Fig. 1, modeling a shipping company. Ignoring the transition guards (shown within squared brackets above transitions), and the read and write operations (denoted by **rd** and **wt** inside the transitions), Fig. 1 shows an ordinary workflow net that consists of 10 places (represented as circles) and nine transitions (represented as squares). There are two distinguished places: `start` and `end`. Place `start` models the initial state of the workflow and `end` the final state.

Initially there is a token in place `start`. The shipper receives goods from a client (transition `receive goods`) to be shipped. In the model, transition `receive good` writes the data of the client (`cl`), the goods (`gds`), and the destination of the goods (`ads`). Then the shipper calculates the price (transition `calculate price`). Afterward, the shipment department and the customer adviser execute their tasks concurrently. If the price of the goods is high (i.e., the transition guard `isHigh(price)` evaluates to true— the exact bound for a price to be high is left unspecified), express shipment is used (transition `ship express`); otherwise, the cheaper standard shipment is used (transition `ship normal`). Based on the same condition, the customer advisor either calculates a bonus (transition `calculate bonus`) for the client and registers this bonus (transition `register bonus`) or no bonus is calculated (transition `no bonus`). In addition, clients sending goods of a high price are notified about their bonus and the shipment (transition `inform by call`); other clients receive only a notification of the shipment (transition `inform by mail`).

As a second contribution, we develop a novel technique for *analyzing soundness of WFD-nets*. The actual challenge is to analyze a WFD-net (i.e., a conceptual workflow) and to conclude from this analysis result whether every data refinement of the WFD-net (i.e., every instantiation of the conceptual workflow with full details) is sound.

First, we study the termination criterion of the soundness property in isolation. Unlike the existing approaches, for example, [4,5], which could give false positives (i.e., the analysis gives the verdict "sound" although the workflow is actually unsound when the data information is refined) or false negatives (i.e., the analysis gives the verdict "unsound" although the workflow with refined data is sound), our technique gives neither false positives nor false negatives.

Our technique is based on the idea of may/must semantics, as introduced by Larsen and Thomsen [6], that guarantees the analysis results to be valid in *any* concrete data refinement. If a WFD-net is proven *must-sound*, then it is sound in any possible data refinement; if it is not *may-sound*, then no data refinement can make it sound. We define the semantics of WFD-nets using the concept of hyper transition systems [7,8] rather than standard may/must transition systems [6]. A hyper transition system allows to connect a single state to a set of successor states (referred to as must-set), thereby increasing the precision of the model. The use of hyper transition systems enables us to improve our previous characterizations of may- and must-soundness from [4]. Moreover, we define the must-sets to be the minimal ones to achieve the most precision possible, thus improving our result from [9]: We show that if a WFD-net is may-sound, there is at least one sound refinement of it, and if a WFD-net is not must-sound, there is at least one unsound refinement.

Example 2. We illustrate may- and must-soundness using the example in Fig. 1. This WFD-net is must-sound; that is, any possible data refinement is sound: Starting with a (colored) token in place `initial`, it is always possible to reach a marking consisting of one token in place `end`. In contrast, if we abstract from data and consider only the underlying workflow net, the net may deadlock. For example, without data the shipment department may decide to use express shipment, but the customer advisor does not calculate any bonus. This yields one token in place `p5` and one token in place `p8`,