Information Systems 34 (2009) 178–208

Contents lists available at ScienceDirect

Information Systems

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

On the relationship between workflow models and document types

Kees van Hee[a], Jan Hidders[b,*], Geert-Jan Houben[c], Jan Paredaens[b], Philippe Thiran[d]

[a] *Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands*
[b] *Department of Mathematics and Computer Science, University of Antwerp, Middelheimlaan 1, BE-2020 Antwerp, Belgium*
[c] *Department of Computer Science, Vrije Universiteit Brussel, Brussels, Belgium*
[d] *Louvain School of Management and University of Namur, Namur, Belgium*

ARTICLE INFO

Article history:
Received 19 August 2006
Received in revised form
28 May 2008
Accepted 23 June 2008
Recommended by: M. Weske

Keywords:
Workflow management system
Petri net
Document management system
Data type
Information system design methodology

ABSTRACT

The best practice in information system development is to model the business processes that have to be supported and the database of the information system separately. This is inefficient because they are closely related. Therefore we present a framework in which it is possible to derive one from the other. To this end we introduce a special class of Petri nets, called Jackson nets, to model the business processes, and a document type, called Jackson types, to model the database. We show that there is a one-to-one correspondence between Jackson nets and Jackson types. We illustrate the use of the framework by an example.

1. Introduction

Data modeling and process modeling are two essential activities in requirements analysis and design of information systems. They are using different techniques and normally they are performed independently. Since both techniques are defining essential aspects of an information system they have to be integrated at some point in the development process, but normally this is at the level of programming. In this paper we show that data modeling and process modeling can go hand in hand from the beginning of the development process of so-called *case-based information systems*. The characteristic of these information systems is that they are developed to support the handling of *cases*, such as the treatment of a patient, the handling of an order or the delivery of a service. For each case type there is a *workflow* defining the tasks to be performed for the case. A workflow is a process with a clearly defined start and end state. In this paper we use a special class of Petri nets to model workflows, the so-called workflow nets [1]. Since in each task of the workflow something happens to the case, it is to be expected that the data type to record the case data is related to the structure of the workflow. The case data is recorded in the *case document*, the structure of which is a *document type*. We show that if we restrict ourselves to a special class of workflow nets, the so-called *Jackson nets*, then there is a tree shaped document type for the case data, called the *Jackson type*, that contains the same information as the workflow net. One of the main results in this paper is that there is a one-to-one correspondence between the document type and the workflow description, so from one we can derive the other. This is similar to the classical program design method of Jackson [2] which is the reason we called the workflow nets and the document types after this author.

The organization of the rest of this paper is as follows. In Section 2 we give the system development context for

* Corresponding author. Tel.: +32 3 2653873; fax: +32 3 2653777.
E-mail addresses: k.m.v.hee@tue.nl (K. van Hee), jan.hidders@ua.ac.be (J. Hidders), geert-jan.houben@vub.ac.be (G.-J. Houben), jan.paredaens@ua.ac.be (J. Paredaens), philippe.thiran@fundp.ac.be (P. Thiran).

0306-4379/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.is.2008.06.003

our work and we give a motivating example. In Section 3 we introduce Jackson types. In Section 4 we introduce the Jackson nets and in Section 5 we study the relationships between Jackson nets and Jackson types. In particular we prove that if two Jackson nets are derived from the same Jackson type they are isomorphic and that if it is possible to derive the same Jackson net from two different Jackson types, these types are algebraical equivalent. In Section 6 we continue with the motivating example. Here we show how we can derive an XML document type from a Jackson net and demonstrate how it provides a logical structure that helps the user to formulate queries over the cases of the workflow. In Section 7 we discuss the usefulness of Jackson types. Finally, we discuss related work in Section 8. Theconclusion of the paper is given in Section 9.

2. Context and motivation

2.1. Historical perspective

In the requirements analysis and design phases of an information system we describe the desired functionality of a system from different perspectives. In the early stages of systems design, say until 1970, the systems designers started to describe the processes the system had to fulfill in terms of *flowcharts*. Since flowcharts describe only sequential processes (one thread of control) the interactions between processes was left out.

In the eighties the *data modeling* techniques became popular. Versions of the *entity relationship* model or the *relational* model were used for this. The big advantage of using this so-called *database-oriented* approach was that after the types of the data stores were established by a data model, several designers could model concurrently the processes that would act on the data stores. The modeling of the *operations*, i.e., of transformations on data objects, was done again at the low level of flowcharts or directly in a programming language.

In the nineties the *object-oriented* approach became popular. In this approach one tries to model the data aspect and the operations on the data in an *integrated* way. However, the processes of a system were still second class citizens. Therefore *process-aware* information systems were identified as special class of systems [3]. This went so far that special software components were designed for the coordination of many interacting processes. Terms as "workflow management", "orchestration" and "choreography" are used to refer to this functionality. Special coordination engines were developed, for instance *workflow management systems*.

Modeling languages for the process appeared. They are also used to configure the coordination engines, like the database schema is a configuration parameter of a database management system. There are two families of formal languages for modeling processes: process algebra's and Petri-nets. Besides these there are several industry standards for modeling processes, such as BPEL (business process execution language) [4], UML activity diagrams [5] and BPMN (business process modeling notation) [6]. These languages allow us to design the process aspect of a system in isolation. These process modeling languages allow concurrency and so the problems of the days of the flowcharts were overcome.

The problem that we address is the integration of the different views: the data view and the process view. Already in the seventies there was a successful attempt to design the data and process aspect in an integrated way, JSP, Jackson's programming method [2] and later the method was lifted to the level of system design, JSD, Jackson's development method [7]. (In *Software requirements and specifications* [8] an overview is presented.) In this approach *hierarchical* program structures where derived from the hierarchical input and output data structures, but they became out of fashion when the relational data model appeared. More recently UML also allows the specification of links between the process models and data models, but these models are here onlyloosely coupled and they remain essentially independent.

The programming method JSP was based on the idea that programs transform data streams into data streams. A data stream was a sequence of data elements and these data streams had a hierarchical data type. In fact, the data types of the input and output streams had to be describable by regular expressions composed of three kinds of operators: *sequential composition*, *selection* and *iteration*. The input and output data types were represented as so-called *tree diagrams* and they were combined into one tree that represented the program structure. In fact, the program structure was also a tree diagram and the input tree and the output tree could be derived from the program tree by projections. The central idea of JSP was that the data structures determine the program structure. So JSP started with designing the input and output data structures. This idea is in line with the database-oriented approach although in JSP hierarchical data structures are essential instead of the relational structure.

The similarity between the Jackson data structures and regular expressions was a reason to compare JSD, the development method based on JSP, with the language for communicating sequential processes, CSP, which can describe regular expressions as well. Therefore Sridhar and Hoare expressed JSD in CSP [9]. To our knowledge this was the first attempt to relate Jackson data structures and process structures in a fundamental way, but there was not much follow up from this attempt.

Another approach to formally integrate processes and data are colored Petri nets where tokens have values that may be changed by transitions [10]. The values are represented as colors and these colors can be linked to edges to indicate that only tokens with a certain color are consumed or produced through them. However, this approach does not offer a way to integrate the types of these colors into a global data model for the process as a whole.

The best practice today in information system development is to model the business processes that have to be supported and the database of the information system separately. This seems to be inefficient because they are often closely related. Like the observations of Jackson, we should try to exploit this relationship as much as possible.