Available online at www.sciencedirect.com
SCIENCE DIRECT®
ELSEVIER
Computers & Industrial Engineering 49 (2005) 199–220
computers & industrial engineering
www.elsevier.com/locate/dsw

A tool for minimizing update errors for workflow applications: The CARD model

Jangha Cho[a], Cheng Hsu[b,*]

[a]*Technical Center, Samsung Electronics Corporation, Suwon City, South Korea*
[b]*Decision Sciences and Engineering Systems Department, Rensselaer Polytechnic Institute, Troy, NY 12180, USA*

Received 22 April 2003; revised 9 August 2004; accepted 10 March 2005

Abstract

Product Data Management, the ASP (application service provider) model of e-business, and other information enterprises involve complex database-driven processes that change frequently. Workflow management for these enterprises requires analysis of the causality of updates on the control logic of the processes and the semantics of the databases involved. Previous workflow analysis tends to focus on the former and leave the details of the database control to application developers. This practice could lead to inconsistent applications when the processes evolve and/or the underlying data and rules change. The field still needs an *integrated* causality analysis tool to facilitate the update problem. In this research, we develop a Control-Activity-Rule-Data (CARD) model of a *decision support tool* that helps workflow administrators analyze the effects of update in processes and/or databases on both control flow and data/rules flow. The CARD model contributes a new Workflow Information Resources Dictionary which represents data, rules, activities and control in a mutually independent but collectively integrated way to achieve the goal. An extended Workflow Causality Graph capable of representing workflow integrity rules enables the CARD model for implementation. An empirical validation of the model using three representative Product Data Management workflow cases at Samsung Electronics Corporation shows its correctness and relevance for practical applications.

Keywords: Workflow management; Causality analysis; Metadatabase

* Corresponding author
E-mail address: hsuc@rpi.edu (C. Hsu).

0360-8352/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cie.2005.03.001

1. The update problem in workflow management: Need to analyze both processes and databases

Workflow management has its origin in the office automation (Bracchi & Pernici, 1984). The attention in the early 1990s on business process re-engineering has further developed workflow management systems into an enterprise integration technology built on a database engine (see, e.g. Derungs, Volga, & Österle, 1997). Today, the trend has encompassed other information enterprises including ERP (Enterprise Resource Planning), PDM (Product Data Management), Supply Chain Management, Customer Relationship Management, and the emerging practice of ASP (application service provider) in e-business (an example of similar observation is given in Sheth, Aalst, & Arpinar, 1999). Workflow management technology has become a major tool for the rapid development and evolution of complex business processes (Cho, 2002).

In essence, workflow management systems first recognize the fundamental workflow tasks (i.e. *activities*) that constitute a business process, and then define the sequencing of these activities (i.e. *control*) for particular instances of the process. Information flow (i.e. *data*) and workflow control knowledge (i.e. *rule*) are encapsulated in the logic of both activities and control and stored in a database. Thus, Activity, Control, Data, and Rule are the four basic elements of workflow we recognize formally in this research (Workflow Management Coalition, 1996, 1999). The problem of workflow updates is concerned with the changes initiated on any of these elements and their immediate cascading effects on other elements. This definition is narrower than some of the previous works by, especially, Ellis, Keddara and Rozenberg, (1995) however, it represents a common and frequent problem in practice. The solution of this problem could facilitate the investigation of the larger problems. We refer to this definition everywhere in the paper when we mention dynamic changes, update effects, and the like.

Despite the advancements in the field (Aalst, Basten, Verbeek, Verkoulen, & Voorhoeve 1998; Aalst, Hofstede, Kiepuszewski & Barros 2003; Bi & Zhao 2003; Casati, Ceri, Pernici, & Pozzi, 1996; Kamath & Ramamritham, 1996; Nutt, 1996), system administrators still lack sufficient tools to analyze update effects and deal with the problem. The reason is simple: while traditional design methods tend to ignore data semantics and leave database structuring to implementation, large-scale and fluid systems such as the above examples nevertheless feature frequent workflow updates that are subject to the encumbrance of the database. In these cases, they need to analyze the effects of updates on both processes and databases in order to minimize errors, not just to analyze processes.

The workflow update problem has two basic technical concerns: how to achieve or retain the *soundness* of the control of *processes* (i.e. initiation, termination, and consistent flow logic), and how to maintain the *integrity* of the *database* objects and inter-relationships (i.e. unique primary keys or object identifications, consistent foreign key values or class inheritances, and consistent data values). We refer to the achievement of both soundness and integrity the *causality analysis* of workflow updates. The reason that updates need integrated causality analysis while the analysis for new system design may not is two-fold. First, updates take place during operation and have to minimize disruption to the operation. Second, updates could take many forms and on any elements of workflow. In contrast, workflow designers could and would follow a hierarchical structure and analyze workflow elements only in sequence: they focus on processes (analyzing activities and control) and leave the detailed analysis of their database implementation (concerning data and rules) to workflow application developers to perform at a later stage. The latter, in turn, would typically hard-code the details into the applications (embedding data and rules in individual activities and control software) without also considering the global interactions of data flow and control flow at the database level. Update analysis at run time, on the