Available online at www.sciencedirect.com

SCIENCE DIRECT®

Data & Knowledge Engineering 50 (2004) 215–240

DATA & KNOWLEDGE ENGINEERING

www.elsevier.com/locate/datak

Algorithms for analysing related constraint business rules

Gaihua Fu [a,*], Jianhua Shao [a], Suzanne M. Embury [b], W. Alex Gray [a]

[a] *Department of Computer Science, Cardiff University, Newport Road, Cardiff CF24 3XF, UK*
[b] *Department of Computer Science, University of Manchester, Oxford Road, Manchester M13 9PL, UK*

Available online 7 February 2004

Abstract

Constraints represent a class of business rules that describe the conditions under which an organisation operates. It is common that organisations implement a large number of constraints in their supporting information systems. To remain competitive in today's ever-changing business environment, organisations are increasingly recognising the ability to evolve the implemented constraints timely and correctly. While many techniques have been proposed to assist constraint specification and enforcement in information systems, little has been done so far to help constraint evolution. In this paper, we introduce a form of constraint analysis that is particularly geared towards constraint evolution. More specifically, we propose several algorithms for determining which constraints collectively restrict a specified set of business objects, and we study their performance. Since the constraints contained in an information system are typically in large quantities and tend to be fragmented during implementation, this type of analysis is desirable and valuable in the process of their evolution.

Keywords: Constraints; Business rules; Constraint analysis

1. Introduction

Constraints are commonly employed in information systems to ensure that the data stored and produced by the systems are valid and represent the intended semantics of the real-world [15,17,25]. Such constraints are typically derived from business requirements, and therefore can be thought of representing a class of business rules that describe the conditions under which an

[*] Corresponding author. Tel.: +44-29-20874000x7329; fax: +44-29-20874598.

E-mail addresses: gaihua.fu@cs.cf.ac.uk (G. Fu), j.shao@cs.cf.ac.uk (J. Shao), sembury@cs.man.ac.uk (S.M. Embury), w.a.gray@cs.cf.ac.uk (W.A. Gray).

0169-023X/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.datak.2004.01.006

organisation operates [18,26]. For example, one constraint might define the criteria that make a customer eligible for a certain kind of discount, which corresponds closely to the organisation's policy on customer discount.

Constraints are typically very volatile, perhaps due to their business origin. New restrictions are frequently imposed on organisations as a result of changes to statutory regulations, government policy and economic/market conditions. Many organisations also choose to modify their business rules on an on-going basis, in order to remain competitive in today's ever-changing business environment. These changes frequently trigger the need to update the constraints implemented in the supporting information systems.

However, updating an implemented set of constraints is not easy. Typically, a constraint begins its life as a high-level (natural language) statement of some condition or policy that the organisation wishes to enforce, and then gets fragmented and dispersed throughout the system in its implementation. For example, a constraint might state that all customers whose total business within the last year is greater than £5000 are eligible for a 5% discount on all orders. The code relating to its implementation might need to be inserted into application programs and the database management system which manage: the creation of new customer orders; modification of existing orders, maintenance of information on current discounts; and archiving of historical customer data. If the mapping from the high-level constraint statement to the software artefacts was accurately documented and maintained, then it would not be very difficult to update this constraint. Unfortunately, many constraints are introduced into an information system as and when they are needed. Even today, constraints are rarely documented comprehensively during system development and any document that is created quickly becomes out of date and unreliable. Therefore, system engineers are often forced to sift through the entire system in order to identify the constraints that are affected by a change made at the high level. This is undesirable because

- First, an information system typically implements a large number of constraints. For example, Herbst et al. reported the existence of 627 business rules in a 12,000 line COBOL application [19], and a similar result was shown in [10], where 809 business rules were found in a 30,000 line COBOL code. Many of these rules are constraints. The large number of constraints makes it a very time-consuming process for the system engineer to manually identify the subset of constraints that is actually relevant to the task in hand.
- Second, without an accurate knowledge of how the statement is mapped to the corresponding software artefacts, trying to determine manually which implemented constraints are affected by the change made to the high-level statement during evolution becomes a rather error-prone process, particularly when the set of implemented constraints is large.

Thus, there is a need for techniques which can help system engineers to locate and piece together the relevant constraints at a higher level automatically. While many theories and techniques have been proposed to assist constraint specification and enforcement in information systems, little has been done so far to help their evolution. In this paper, we introduce a form of constraint analysis that supports constraint evolution. More specifically, we assume that the set of constraints implemented in an information system has already been identified or recovered, either from available documentation, domain experts or through reverse engineering [10,12], and we propose several algorithms for determining which subset of the constraints collectively restricts