The Journal of Systems and Software 82 (2009) 915–931

Contents lists available at ScienceDirect

The Journal of Systems and Software

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

# Lightweight query-based analysis of workflow process dependencies

W. Dai [a], D. Covvey [b], P. Alencar [a,*], D. Cowan [a]

[a] University of Waterloo, David R. Cheriton School of Computer Science, Waterloo, Ontario, Canada N2L3G1
[b] University of Waterloo, Department of Biology, School of Optometry, Waterloo, Ontario, Canada N2L3G1

ARTICLE INFO

*Article history:*
Received 27 November 2006
Received in revised form 27 November 2008
Accepted 8 December 2008
Available online 3 February 2009

*Keywords:*
Software process
Workflows
Software change
Dependency analysis
Logic programming
Software maintenance

ABSTRACT

Dependency analysis is important in all of the stages of workflow processes. Workflow elements and features are always difficult to track and as their changes often lead to code tangling as a result of the addition of new requirements and programs. This mosaic quality complicates program comprehension and maintenance. Therefore, an appropriate analysis will help us to identify the potentially affected entities if changes occur. In this paper we present an approach for the lightweight analysis of workflow process dependencies, which include routing, data and roles dependencies. The model is represented as a knowledge base using a logic programming language, Prolog. We develop a set of query rules that can be applied to the well-defined knowledge base at both activity and process levels to retrieve the potentially affected entities. Finally, we use a case study of workflow processes in the healthcare domain to show how our dependency analysis approach works.



## 1. Introduction

Like any other system, a workflow process is composed of different kinds of components or entities. These entities play different roles in a workflow process and they also interact with each other in all aspects. That is, entities in a system are not independent of each other and there always are relationships among these entities directly or indirectly. One of the most commonly identified relationships is a dependency relationship, which means an entity depends on other entities. This fact naturally leads to a requirement in system's analysis, i.e., dependency analysis.

Our work in this paper is about the change impact analysis of workflow process through dependency analysis. Our contributions include a multi-dimensional workflow process dependency model that involves critical aspects of dependency relationships existing in workflow processes: routing, data and role dependencies. We then formally represent this model by a logic programming language, that is, Prolog, through a well-defined knowledge base. Finally, we develop an effective tool-supported approach, the use of query rules, to identify the potentially affected entities in the process analysis domain.

### 1.1. Motivation and related work

In the real world, there are numerous causes of changes to existing workflow processes (Allen, 2001; Workflow Management Coalition, 1999, 1995, http://www.wfmc.org). For example, the modification of laws or regulations may require the workflow process to be changed to comply with the new laws or regulations; new medical or healthcare knowledge may require healthcare providers to make corresponding changes in their health service process implementations; new technologies may be introduced into business processes (Ader, 2003; Kappel et al., 2000; Loos and Fettke, 2001; Reijers, 2003), which replace the jobs previously done by humans to increase efficiency; new systems may be deployed to upgrade old ones in current workflow processes and in the context of workflow management systems (Becker et al., 1806; Chountas et al., 2003; Cicekli, 1999; Endl and Meyer, 1999; Kafeza et al., 2001; Millie Kwan and Balasubramanian, 1997; Mohan et al., 2002; Ren et al., 1999), and so on.

Regarding change, typically it can occur in several states of a process life cycle: during the design, reengineering or redesign, and maintenance of the process. During the design stage, change is usually driven by the customer requirements, as typical in system and software design domains. On the other hand, after a process is designed and deployed in a specific domain, sometimes the process may need reengineering (e.g., process reengineering (Fitzgerald and Siddiqui, 2002; Weerakkody and Currie, 2003)). Further, in order to make processes satisfy new expectations and requirements such as efficiency and effectiveness, modifications

* Corresponding author. Tel.: +1 519 8851211x33944.
*E-mail addresses:* wdai@cs.uwaterloo.ca (W. Dai), dcovvey@csg.uwaterloo.ca (D. Covvey), palencar@cs.uwaterloo.ca (P. Alencar), dcowan@csg.uwaterloo.ca (D. Cowan).

0164-1212/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2008.12.050

to an existing process may be required during workflow process maintenance.

As a result of changes to workflows, there could be impacts on other activities and processes if some worflow activities or processes are modified. For example, if data produced by a preceding activity are for any reason not available, the activities depending on this activity cannot be activated. This may lead to the whole process not being executed.

Since these changes will continue during the entire process life cycle, an analysis is needed when we handle the impacts of changes at different stages and levels. Moreover, given these examples, and the fact that there often is a complex dependency relationship among workflow processes and activities, we think a workflow Process Impact Analysis (PIA) is an effective way to predict what kinds of impacts we will have. Through PIA, we can identify the affected activities and processes and adapt our process appropriately to satisfy new requirements. In addition to this, new business and workflow processes are being deployed everywhere in order to deliver competitive services to customers or clients. Our objective is to help organizations save effort and time, where the alternative can be business failure. It is for this reason we propose dependency analysis.

### 1.2. Related work on impact analysis

Workflow process research has existed for more than 30 years and is gaining increasing attention from both industry and academy. However, we find that little work has been done on workflow process impact analysis. Dai et al. (2004), Reijers (2003) have categorized current workflow process research into the following areas: workflow modeling and validation, workflow performance analysis, and process reengineering. Another area of workflow research addresses the design of workflow management systems, which provide runtime environments for workflow execution. Even the dependency relationships within and among workflow processes have been identified in Ajila (1995), Adam et al. (1998), Chountas et al. (2003), Chun et al. (2002), Eder et al. (1999a), Eder et al. (1999b), and Kappel et al. (1998). However, these authors focus on workflow modeling or representation, and most of their dependency relationships are limited to process structure dependency, i.e., control or routing dependency. We also notice that these structure dependency relationships are limited to intra-dependency without consideration of inter-dependency, as in our example scenario. We can see the limitation of focus to process structure dependency leads to an incomplete understanding of dependencies. In another paper, Kim (2003) introduces a dependency analysis framework consisting of four separate dependency nets, i.e., activity, role, data and actor. However, the goal of this framework is not to deal with changes to processes and the analysis of their impacts, but rather to generate a set of "transition conditions" that later are deployed in a distributed workflow enactment system, i.e., a management system for process execution.

Although there are few references for workflow impact and dependency analysis, we can take advantage of methodologies and techniques widely applied in software impact analysis, as we realize that a workflow process and a software system share many common characteristics. For example, a software method built on another one is like a complex activity composed of simple ones; interaction between different applications or classes is like interactions among activities or processes; a branch path in code is like parallel control flow; and software application deployment and execution is like process deployment and execution. Actually we can treat a workflow process as a variant of a software application. From this point of view, we believe that the workflow process impact analysis deserves the same attention in workflow process research as impact analysis in software research.

#### 1.2.1. Software impact analysis

Software impact analysis is often used to assess the effects of a change on a system after that change has been made. However, a more proactive approach uses impact analysis to predict the effects of change before it is instantiated (Bohner, 2003; Boher and Arnold, 1996). In fact, currently the main goal of impact analysis is to identify the software products and entities affected by proposed changes and to produce a list of entities that should be addressed during the proposed change process. As a result, we can evaluate the consequences of planned changes as well as the trade-offs among the approaches to implement the changes. Finally, we decide whether or not to make the changes, or we find other ways to make the changes based on our evaluation.

Most researchers follow the partitions in Boher and Arnold (1996) which classifies impact analysis techniques into two broad categories: dependency analysis and traceability analysis. For dependency analysis, tools are developed to detect and capture dependency information in the system source code artifacts. It includes three subcategories: data, control, and component dependency relationships. Among these relationships, data dependencies are relationships among program statements that define or use data. That is, the data dependence exists when a statement provides a value used by another statement in a program. Control dependencies are relationships among program statements that control program execution, while component dependencies refer to the general relationships among source-code components such as modules and files. Traceability analysis generally is manual work that focuses on modeling dependencies from the perspectives of software engineering environments and documentation systems that contain varied levels of software information. Examples include requirements traceability that identifies parts of the software that may change with changed requirements, software documentation traceability that identifies the component relationships through the use of a document repository, and project information database traceability that provides database-query mechanisms to help software engineers determine the potential impacts of changes based on the project management database. Some document management systems have been developed to assist this analysis, e.g., document browsers. Although dependency analysis (Ajila, 1995; Deruelle et al., 1873; Fisler et al., 2005; Law and Rothermel, 2003; Moonen, 2002; Ryder and Tip, 2001) and traceability analysis (Arango et al., 1993; Baniassad and Clarke, 2004; Knethen, 2001; Marcus and Maletic, 2003) can be used separately, they also can be used together to achieve an analysis goal as in Lindvall and Sandahl (1998).

#### 1.2.2. Dependency analysis

As an effective and proven methodology, dependency analysis has been the most mature technique available in impact analysis. The dependency relationships are typically represented as graphs or tables that assist people in understanding the dependencies. Dependencies are stored in a dependency graph in which usually each node represents an entity and each directed edge represents a dependency relationship between entities (Boher and Arnold, 1996). On the other hand, we can see that the procedure to identify the affected entities, the types of changes, and the effects of a change, is time consuming and costly if handled manually as the analysis domain may contain many entities and have various dependency relationships at different levels. To deal with this problem, researchers have developed various query or lookup mechanisms that usually are associated with the dependency representation, which provides a foundation for the query. These query mechanisms enable users to select the types of dependencies to be analyzed from the identified dependency relationships and infer the affected entities (Ajila, 1995; Boher and Arnold, 1996; Deruelle et al., 1873; Fisler et al., 2005; Robillard and Mur-