Automation in Construction 10 (2001) 639–643
www.elsevier.com/locate/autcon

Agents and suggestions in a Web-based dynamic workflow model

Malcolm Crowe [a,*], Sandy Kydd [b]

[a] Department of Computing and Information Systems, University of Paisley, Paisley, Scotland PA1 2BE, UK
[b] DMC Ltd, 3 La Belle Place, Glasgow, UK

Abstract

Two features of this dynamic workflow system make it suitable for the use of quasi-intelligent agents: (a) workflow processes need not be fully specified, and so can be non-prescriptive in approach, and (b) a job can be modified independently of the process of which it is an instance, and so some participants may have permissions to change its course. In the architecture that has been chosen for the research, web clients are used, and web agents generate suggestions based on analysis of the process itself, the current job, and the records of previous jobs. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Agents; Suggestions; Dynamic workflow

1. Introduction

In today's Internet-enabled society, many applications that would normally have been restricted to only being available within an organisation are now online, and can span not only companies and organisations, but also national boundaries, to be become truly location-independent. Workflow systems, where participants in a workflow process may not be contained solely within one organisation and/or location, are prime examples of applications that can take advantage of this new ubiquity in computer connectivity. The Internet, and consequently, the World Wide Web as its most user-friendly incarnation, is an obvious arena in which workflow systems should become established.

The term dynamic workflow means that the elements defined in a workflow process are not defined rigidly, and can be changed or altered to reflect the way that instances of the workflow process are actually run. The workflow process is dynamic — although elements in it are defined, they can be changed when a job is run for the process. Elements can be edited for individual jobs, and exception conditions handled. A logical extension of this model is to allow process definitions to evolve based on what is actually being done in jobs for the process, when users make changes in the jobs to the definitions initially supplied from the process. This leads to the idea of automatically suggesting changes to a process definition in a process review phase, based on analysis of the jobs that have been run for the process.

The basic architecture for the prototype system is that of a workflow database accessed via various programming interfaces, and web-based user clients, other client applications and standalone agent clients

* Corresponding author. Tel.: +44-141-848-3300.
E-mail address: crow-ci0@paisley.ac.uk (M. Crowe).

manipulating the database contents through the defined interfaces. The client applications allow the creation of processes, the instantiation of jobs from the processes, and running of these jobs. The agent applications generate suggestions in the database.

In the dynamic workflow system prototype, the server side scripting language PHP is used to generate dynamic content from the database for producing the standard web pages in the system [3]. PHP provides a simple scripting language that can be embedded in specialised tags in HTML documents. When a user requests such a document, the document is parsed by the PHP interpreter prior to sending it back to the web client, and the embedded scripts can generate the dynamic portions of the document at this point. The main reasons this language was chosen is that the scripting language is fast, simple, and it is easily interfaced to a large number of SQL database systems. The initial implementation was attempted using Perl as the language to generate the dynamic content as a standard CGI application [4], but PHP proved to have a large performance advantage in generating the dynamic content, especially when used as a module for the Apache web server [2]. It was also a lot simpler for making changes to the web pages in the user interface. However, certain portions of the web interface, namely the workflow calculation, were retained as Perl CGI scripts, since the activity rules were specified in Perl.

Since the prototype workflow system was built as a web application, the user interface to the system is a web browser, and the server parts of the system, including the database, were interfaced through the web server. The other client parts of the system (i.e. the suggestion generator agents) were implemented on the server platform for the sake of simplicity, although this is an implementation decision, rather than a restriction in the system architecture. The suggestion generator agents and other utility programs were written in Perl, and this language proved to be fast and flexible for producing such ad hoc applications.

2. Suggestions

Suggestions are workflow elements that were not created in the process definition, but are produced by the workflow system based on analysis of jobs that have been instantiated and run from a process definition. The purpose of these suggestions is twofold:

- to produce revisions to the definition of a process encapsulated within an individual running job (i.e. the changes to the job definition)
- to produce revisions to the process definition for all subsequent instantiations of the process

Ref. [1] talks about “suggesting revisions to the process plan”. This refers only to the first case mentioned above, i.e. the “runtime enactment of a process by multiple participants”. The dynamic workflow model described here extends this to include the second case as well. In this way, the workflow system cannot only generate suggestions that will be used in an individual job, but it can analyse past jobs run from a process definition and suggest ways in which the process definition could evolve to reflect the way that the jobs were actually run.

Suggestions for a job can be generated for a job on a per-job basis, by analysing the activities and links in a job definition that have not yet been enabled and executed, and trying to guess which activities might be appropriate for participants to do next, even if they have not been scheduled by the formal workflow definition. These are classified as lookahead suggestions. Lookahead suggestions are seen as necessary in providing support for exception handling, as they can be generated to enable activities that have not been defined to be executed in the job, but which are only invoked when some constraint condition is broken.

A simple method of generating lookahead suggestions for a job is given below:

1. Get the activities for the job that have not yet been run
2. For each activity, evaluate its pre-condition, and if it is true (i.e. the activity has met its pre-conditions for execution), create a suggested work item

Using lookahead suggestions means that some of the constraints of traditional workflow systems in having temporally linear execution of activities can be overcome. For example, two activities may be