A Specification Language for the WIDE Workflow Model *

Daniel K.C. Chan† Jochem Vonk† Gabriel Sánchez‡ Paul W.P.J. Grefen† Peter M.G. Apers†

†Computer Science Department, University of Twente, 7500 AE Enschede, The Netherlands
‡Sema Group sae, c/ Albaracin 25, 28037 Madrid, Spain
{chan, vonk, grefen, apers}@cs.utwente.nl, gsg@sema.es

Abstract

This paper presents a workflow specification language developed in the WIDE project. The language provides a rich organisation model, an information model including presentation details, and a sophisticated process model. Workflow application developers should find the language a useful and compact means to capture and investigate design details. Workflow system developers would discover the language a good vehicle to study the interaction between different features as well as facilitate the development of more advanced features. Others would attain a better understanding of the workflow paradigm and could use the language as a basis of evaluation for the functionality of workflow systems.

1 Introduction

There is a consensus that fundamental workflow management systems [7] need to be extended to meet the requirements of production workflow systems [5] and novel application domains [3]. Studies and discussions of such extensions have been hampered by the lack of a concrete framework. The aim of this paper is to introduce a concrete framework through the presentation of a workflow specification language. The language differs from other workflow languages and proposals in that it captures all the fundamental elements of the workflow paradigm (namely organisation model, information model, process model, and their relationships) at an abstraction level that is suitable for user development rather than machine manipulation. In other words, it provides a higher level of abstraction than many existing workflow languages.

The provision of a textual description for workflow applications is important for a number of reasons. If a workflow system is to be around for a considerable length of time, it may well require upgrading every now and then. Given the rapid development in workflow products, upgrading may be realised by switching to a different implementation platform or workflow engine. A specification language can ease the transition in both circumstances by providing a bridge between different systems. Besides being part of a workflow system, a specification language can also be used for process documentation.

The specification language presented in this paper has served as a vehicle for the development and synthesis of advanced workflow concepts within the ESPRIT WIDE project (see http://www.sema.es/projects/WIDE/ for further information) [2]. The aim of the project is to deliver a commercial strength advanced workflow system to replace the FORO workflow system currently marketed by the Sema Group.

2 An Overview of the WIDE Model

The WIDE workflow model, which is inspired by the workflow model used in FORO, is an extension of the reference workflow model proposed by the Workflow Management Coalition [7]. Unlike the reference workflow model, it supports a rich organisation model, sophisticated activity assignment constraints, dynamic control flows including the use of active rules, complex process structures, and workflow transactions. A summary of the three "sub-models" of the WIDE workflow model is given next. A complete description of the WIDE model can be found in [1].

Organisation model. It registers the organisation structure and resources of an enterprise. It records information about individual employees (*staff*), functional positions held by staff (*position*), groups of staff (*team*) that are put together to serve some business transaction, as well as non-human resources (*tool*) such as machines and software. A staff member can hold several positions as well as participate in a number of teams possibly in different capacities according to the positions held. Staff, positions, teams, and tools are collectively referred to as *agents*. All agents are associated with a *domain* which is usually used to model geographical sites or functional units of an organisation. The *deputy* relationship allows one staff member to deputise for another that is not available to carry out an activity. The *position hierarchy* captures the *accountability* relationship between positions. When a staff member is not available, a task can be performed instead by his or her senior to whom the staff member is accountable for. The *team hierarchy* recording the *inclusion* relationship serves a similar purpose. When a team member is not available, the team leader can take over. However, if no one in the team is available, the leader of the *affiliated* team which includes the original team can be brought in.

Information Model. It defines the data used in a workflow process, governs the operations that can be performed on the data, as well as controls the scope and presentation of the data. Data can have one of two possible scopes: *global* and *local*. Global data are shared by all workflow processes. They are persistent and are stored usually in either databases or files. Local data are only accessible from within one workflow process and are shared between activities within the same process. The *entity* data type is introduced to capture external data and their associated operations. Every entity may have a set of associated operations that are often completely different from any other entities. The *form* data type is probably the most distinguishing type in the information model. It resembles the record type but with presentation information which is important due to the interactive nature of workflow systems. Each field in a form is typed and can be used to show or update data variables. Default field values can be defined. A form type can be derived from another form type possibly by removing fields. Moreover a derived form can relax or restrict operations that can be performed on a field. This mechanism encourages and supports re-usability which reduces the maintenance effort.

Process Model. The use of activity abstraction in the process model allows an application to be defined using top-down refinement which makes the ordeal more manageable and the resultant design more comprehensible. In other words, an activity representing a unit of work at one level can be refined into a network of activities in the next level. Activities at the bottom level, so to speak, are the building blocks of the process where actions are actually carried out. The bottom-level activities are referred to as *base tasks* while other activities are called *tasks*. This notion of activity abstraction is exploited in the transactional support within the WIDE model [6]. Every activity definition can contain the following components: (1) a *pre-condition* that has to be satisfied for the activity to start, (2) actions performed by the activity, (3) a *post-condition* that has to hold for the activity to terminate, (4) a *role constraint* about assigning the activity to an agent and making it observable to an agent, (5) a *schedule* for the execution of the activity and its priority, and (6) handlers for system-defined as well as user-defined *exceptions*. The action part of an activity varies depending on whether the activity is a task or a base task. For a task it captures the control flows between activities in the next level. For a base task it captures some computation which may be performed over the information model or some external operation performed by the user. Consequently, role constraints may differ between tasks and base tasks. For a task a role constraint may specify dependency between activity assignments of lower-level activities. For a base task a role constraint is always about the base task alone. Base tasks have the option of specifying the forms that can be used for interaction with the user.

3 Example Specifications

The example used in this section is about an hypothetical application about publishing technical papers produced within the WIDE project. Before a paper is submitted for publication, an internal review is carried out. If the result of the review is favourable, the paper will be submitted. Otherwise, the paper will be rewritten and reviewed again. The rewrite and review cycle can iterate for a number of times before the paper is submitted. Specification of forms and processes is presented here. Specification of data and organisation is not presented here but can be found in [4].

```
1  FORM_MODEL Paper_Review OF CS OF Twente
2    FORM Review
3      HEADING "REVIEW FORM"; SPACE;
4      LABEL "Paper" LINKS ENTITY: paper;
5      LABEL "Number" LINKS INTEGER[3] : number; NEW_LINE;
6      LABEL "Title" LINKS STRING[120] : title; NEW_LINE; SPACE;
7      HEADING "Rating (1-6)";
8      LABEL ["Originality", "Technical Quality", "Significance",
             "Presentation", "Overall Rating"];
9      LABEL LINKS ARRAY[5] OF INTEGER[1] : ratings; NEW_LINE;
10     SPACE; HEADING;
11     LABEL "Comment" LINKS STRING[1000] : comment; NEW_LINE;
12     LABEL "Reviewer" LINKS STRING[20] : reviewer;
13     LABEL "Date" LINKS DATE: review_date;
14   END_FORM;

15   FORM Author_Form DERIVED_FORM Review
16     WITHOUT reviewer;
17     ONLY READ FOR ratings, comment, review_date;
18   END_FORM;

19   FORM Reviewer_Form DERIVED_FORM Review
20     ONLY READ FOR number, title;
21     ONLY print FOR paper;
22     INITIALLY
23     ratings.[1] = NULL, ratings.[2] = NULL,
24     ratings.[3] = NULL, ratings.[4] = NULL,
25     comment = NULL, reviewer = NULL, review_date = NULL;
26   END_FORM;
     ...
27 END_MODEL
```

Figure 1: Form Model - Publication Process

Form Specification. A form specification contains representation details concerning the display of a form and the types of the data that can appear in a form. A typical example is the "Review" form defined in line 2-14 in Figure 1. The heading statements (line 3,7,10) mark a new formatting section and optionally print a string on the screen (line 3,7). While NEW_LINE and SPACE correspond to starting a new line and producing some vertical spacing. A LABEL statement usually generates a number of labelled boxes (or menus) though either the boxes (line 8) or the labels (line 9) can be omitted. It should be noted that fields in a form are references to local data and they are not value holders themselves.

The "Author_Form" is derived from the "Review" form and differs from the latter in two ways. First, the field "reviewer" is removed (line 16). Second, the fields "ratings", "comment", and "review_date" cannot be changed (line 17). Similarly, "Reviewer_Form" is derived from the "Review" form. It restricts access to field "paper" to only the user-defined operation "print" (line 21). Initial values are also given for other fields (line 23-25) where NULL is a common value for all types. Derived forms are therefore used to control the scope and access of local data by means of structural changes, access constraints, and initial values for fields before the form is displayed.

Process Specification. The publication process be-

```
1  WORKFLOW_MODEL Publishing OF CS OF Twente
2  USES ORGANISATION_MODEL WIDE;
3  USES FORM_MODEL Review_Form;
4  USES DATA_MODEL Publication;
5  CONTROL
6    START Submit_Paper;
7    Submit_Paper ENABLES Receive_Review;
8    Receive_Review IF p_accepted = TRUE
9                ENABLES Polish_Paper;
10   Receive_Review IF p_accepted = TRUE
11               ENABLES Solicit_Funding;
12   END Receive_Review IF p_accepted = FALSE OR
13         (Polish_Paper AND Solicit_Funding);
14 ROLE
15   Submit_Paper.Write_Paper,
16      Polish_Paper.Write_Paper BY SAME STAFF;

17 TASK Submit_Paper
18   START Write_Paper;
19   Write_Paper ENABLES Review_Paper;
20   Review_Paper IF p_ratings.[4] > 3 ENABLES Send_Paper;
21   Review_Paper IF p_ratings.[4] ≤ 3 ENABLES Write_Paper;
22   END Send_Paper;
23   ROLE
24     Write_Paper, Review_Paper BY DIFFERENT STAFF;
25 END_TASK;
     ...
26 TASK Polish_Paper
27   START Write_Paper;
28   Write_Paper, Solicit_Funding TOGETHER ENABLE Send_Paper;
29   END Send_Paper;
30 END_TASK;
   ...
31 BASE_TASK Review_Paper
32   VIEW Reviewer_Form(p_paper, p_no, p_title, p_ratings,
                       p_comment, p_reviewer, p_date);
33   PRE_CONDITION
34     Reviewer_Form.paper <> NULL;
35     Reviewer_Form.number <> NULL;
36     Reviewer_Form.title <> NULL;
37   MESSAGE
38     "Please fill in all fields.";
39   POST_CONDITION
40     EVERY Review_Form <> NULL;
41   ROLE
42     BY SOME x STAFF PROVIDED "english" IN x.languages;
43 END_TASK;
   ...
44 END_MODEL
```

Figure 2: Process Model - Publication Process

gins with importing an organisation model (line 2), a form model (line 3), and a data model (line 4) relevant to the application. The control part, line 5-13, specifies the top level activity abstraction. The workflow starts with the "Submit_Paper" task (line 6) and finishes with either the "Receive_Review" task (line 12) or the two tasks "Polish_Paper" and "Solicit_Funding" (line 13). The transition from one task to another is captured using the ENABLE statement (line 7 shows its simplest form). Conditional transitions can be specified as in line 8 and 10. The task "Polish_Paper" is refined in line 26-30. The *total join* from "Write_Paper" and "Solicit_Funding" to "Send_Paper" is captured using the keyword TOGETHER ENABLE.

In the case of a base task, instead of control statement, computation and message display are specified (line 37-38). The message may be displayed together with other forms. Forms are displayed only by base tasks which have the duty of performing computation that may require inputs from the user. The forms to be displayed is specified in the VIEW clause and their contents are filled using local variables (line 32). Every activity can be given a pre-condition and a post-condition that have to be satisfied for it to begin and end successfully. The pre-condition of "Review_Paper" given in line 33-36 specifies that some fields in the form must be filled. The post-condition given in line 39-40 specifies that all the fields must be filled. The keyword EVERY is a shorthand for all the fields in a form.

Activity Assignment. Role constraints are used to govern the assignment of activities to agents. Constraints defined in base tasks reflect enterprise-wide policy for the activities. The one given in line 41-42 in Figure 2 specifies that base task "Review_Paper" is to be performed by someone who knows "english". When activities are put together to form an application, additional constraints are usually required to dictate relationships between activities. The constraint specified in line 23-24 insists that the author cannot review his or her own paper while the constraint in line 14-16 requires that the person who writes the paper to be responsible for its polishing.

4 Conclusions

This paper presented the WIDE workflow model and a user-oriented specification language for the model. The specification language differs from existing workflow specification languages and proposals in that all fundamental elements of the workflow paradigm are captured and elegantly integrated. Distinctive features include use of possibly partially parameterised domains, combination of specifications, incorporation of presentation details in form specifications, reusability over form specifications, declarative activity assignment, declarative exception handling, and workflow transactions. The specification language forms the basis of the current WIDE specification language whose implementation has just been completed.

References

[1] F. Casati, P. Grefen, B. Pernici, G. Pozzi, and G. Sánchez. WIDE Workflow Model and Architecture. Technical Report 96-19, Centre for Telematics and Information Technology (CTIT), University of Twente, Netherlands, 1996.

[2] S. Ceri, P. Grefen, and G. Sánchez. WIDE - A Distributed Architecture for Workflow Management. In *Proceedings of the International Workshop on Research Issues in Data Engineering*, pages 76–79. IEEE Computer Society Press, 1997.

[3] D.K.C. Chan and K.R.P.H. Leung. A Workflow Vista of the Software Process. In *Proceedings of the International Workshop on Database and Expert Systems Applications*, pages 62–67. IEEE Computer Society Press, 1997.

[4] D.K.C. Chan, J. Vonk, G. Sánchez, P.W.P.J. Grefen, and P.M.G. Apers. A Conceptual Workflow Specification Language. Technical Report 96-48, Centre for Telematics and Information Technology (CTIT), University of Twente, Netherlands, 1996.

[5] D. Georgakopoulos and M. Hornick. An Overview of Workflow Management: from Process Modeling to Workflow Automation Infrastructure. *Distributed and Parallel Databases*, 3(2):119–153, 1995.

[6] P. Grefen, J. Vonk, E. Boertjes, and P. Apers. Two-Layer Transaction Management for Workflow Management Applications. In *Proceedings of the International Conference on Database and Expert Systems Applications*, volume 1308 of *Lecture Notes in Computer Science*, pages 430–439. Springer-Verlag, 1997.

[7] Terminology & Glossary. Technical Report TC-1011, Workflow Management Coalition, June 1996.