Simple models for articulating complex work processes

Alessandra Agostini, Giorgio De Michelis

Cooperation Technologies Laboratory, DSI - University of Milano, Italy

{agostini, gdemich}@dsi.unimi.it

Abstract. Adaptability of workflow models can be achieved if the specification module of the workflow management system is based on a theoretical kernel, allowing minimal effort for the creation of a workflow model capable of providing, when necessary, any view (and the related services) the users may want of it. The specification module of the workflow management system of the Milano prototype has been developed on the basis of the above assumption. In this paper we show how it allows to support the articulation of complex work processes, with particular attention to exception handling and static and dynamic changes.

1. Introduction

Work processes consist, generally, of interdependent and distributed activities, performed by cooperating actors. To prevent them from degenerating into chaos, the cooperative effort of their actors must be coordinated, aligned, integrated, meshed - in short, articulated [Schmidt, Simone, 1996]. Articulation sometimes is local to a single work process, sometimes involves a group of work processes distributed in space and time (in this case the first act of articulation is considering them as instances of a unique work process type or class). The fundamental way a single work process can be articulated is through enforcing the mutual awareness of the actors of what is going on in the process. As it is frequently the case, the degree of complexity of a single work process and, more frequently, of a group of work processes, is very high, and therefore mutual awareness of the actors cannot be reached without the use of some social artifacts that are built for coordination purposes as, e.g., forms and workflow models (both maps and scripts; Schmidt, 1997). The main role of these coordinative artifacts is to provide simple means to abstract from complex interdependencies of the work process (instances) for articulating it (them).

Let us restrict our attention to workflow models and instances of one work process class. A workflow model fixes the standard and/or "preferred" causal dependencies (data and/or control flow) between the activities of all the instances of the same class, so that the actors are helped in understanding their situation and in acting effectively. This means, for example, that managers can control the ongoing instances with respect to the standard flow described by its workflow model for evaluating their efficiency and their correctness; that the performer of an activity knows what she has to do and when, looking at the activity description in the workflow model, without taking into account what the other actors are currently doing; that the customer of the service provided by the work process can monitor its execution making reference to the workflow model for foreseeing its completion time; and finally that, when a breakdown occurs during the execution of an instance, the performers can share their understanding of the situation and co-decide their moves to deal with it, looking at the workflow model.

When a workflow model is embodied into a computer-based system, such as a workflow management system, it supports the automation of the standard flow of work, while it provides users with the necessary information to deal with breakdowns (Ellis et al., 1995; van der Aalst, 1998; Borgida, Murata, 1998; Agostini, De Michelis, 1998). Effective workflow models, therefore, are both executable codes and cognitive artifacts.
A coordinative artifact, as a workflow model, looses its effectiveness when its use becomes as complex as the direct articulation of the interdependencies it abstracts from, either because it is too complicated or because it does not reflect (model) properly them. In the first case, in fact, using it for understanding the situation of the ongoing work process instance(s) and/or changing it (them) can become per se a complex work process needing specific coordinative artifacts to be articulated; in the second, the actors, instead of being supported in reaching mutual awareness disregarding the interdependencies it abstracts from, are frequently forced to invent together ways for circumventing it, in order to perform effectively (Bowers et al., 1995).

On the other side, a good model should provide a large variety of services: the standard workflow should be distinguished and supported by automatic coordination of its activities; the ongoing instances should be tracked on the workflow model in order to support accounting and planning; breakdowns should be supported by exception handling mechanisms to be activated within the interactions the actors perform to share their understanding of the situation; changing the workflow model should be supported by automatic correctness checks; automatically enacting changes on the ongoing instances should be allowed and prevented from failures.

Is it possible to have all these services by a unique simple workflow model? Architects and civil engineers are today supported by CAD systems using which they can build a three dimensional model of a building from a minimal set of two dimensional components, receiving back any sort of two- and three dimensional representations of it. Projective geometry is the theoretical kernel of those modeling systems. The specification module of a workflow management system should as well be based on a theoretical kernel, allowing minimal effort for the creation of a workflow model capable of providing, when necessary, any view (and the related services) the users may want of it.

We have developed the specification module of the workflow management system of the Milano prototype on the basis of the above assumption (Agostini et al., 1997; Agostini, De Michelis, 1998). In this paper we show how it allows to support the articulation of complex work processes. An example derived from a real credit procedure used within an Italian bank is used throughout the paper.

2. The Workflow Management System of Milano

In 1994 at the Cooperation Technologies Laboratory the authors -together with Maria Antonietta Grasso and several students- started the development of the prototype of a new CSCW system, called Milano (De Michielis, Grasso, 1994; Agostini et al., 1997).

Milano is a CSCW platform supporting its users to keep themselves aware of the history they share with the actors with whom they cooperate and of the activities they are committed to perform in the future. The perspective from which we observe work practices can be considered a Situated Language-Action Perspective (Suchman, 1987; Winograd, Flores, 1986; De Michielis, Grasso, 1994).

Milano integrates in the user workspace a workflow management system (MWMS) with a multimedia conversation handler (MCH), and, finally, an object repository (MOR) where an organizational handbook is stored.
Without adding more details about the other components of Milano—the interested reader can find a more complete account on it in (Agostini et al., 1997)—let us concentrate on the main characteristics of its workflow management system, paying particular attention to its specification module.

The workflow management system of Milano is a new generation workflow management system (Abbott, Sarin, 1994): its aim is to support its users not only while performing in accordance with the procedure described in its model, but also when they either need to follow an exceptional path or when they need to change the workflow model. Within Milano a workflow model is an important part of the knowledge its different users (the initiator of a workflow instance, the performer of an activity, the supervisor of the process and, finally, the designer of the workflow model) share while cooperating within a work process.

The workflow model is not merely a program to be executed and/or simulated by the execution module with a graphical interface to make it readable by its users. Rather, it is a formal model whose properties allow the user to get different representations of the workflow, to compute exceptional paths from the standard behaviour, to verify if a change in the model is correct with respect to the target of the workflow and to enact safely a change on the ongoing instances.

The specification module of the Milano workflow management system is based on the theory of Elementary Net Systems (ENS) (Rozenberg, 1987; Thiagarajan, 1987). Moreover, since Milano is based on the idea that workflows must be as simple as possible, its workflow models constitute a small subcategory of ENS, namely Free-Choice Acyclic ENS, whose main properties are computable in polynomial time, allowing an efficient realization of the specification module. The class of Free-choice Acyclic ENS has some nice mathematical properties that appear suitable to provide the above services. Shortly, there is a synthesis algorithm from Elementary Transition Systems (ETS) to ENS (Nielsen et al., 1992) so that the user can have different views of the workflow (based on local conditions or global states; see Figure 2); it is possible to compute and classify forward- and backward-jumps linking their states; morphisms between models allow an efficient consistency check for both static and dynamic changes. The reader can find the details of the formal aspects of the Milano workflow management system in (Agostini, De Michelis, 1998).

3. Workflow models of a Credit Procedure

The case we discuss in this paper is the process through which a customer request for a new credit is managed by a bank. This example is extracted from a real case study; for a more complete description see (Schael, Zeller, 1993; Agostini et al., 1994).

In the credit procedure the client interacts with the Agency Director (AD) who is responsible for the whole procedure. After a preliminary informal investigation, where the client motivation is exploited, the documents the clients provides to support her request are collected and two parallel processes start: in the first, the information about the client accounts are collected and the practice is perfected, while in the second several external data bases are checked to control if the client has any
insolvency or any other critical financial situation. The two processes join when a report on the credit request is written. At this point the decision process starts, that follows different paths depending on the value of the requested credit. For instance, if the credit request is under 50,000,000 Lit. the AD can write a credit proposal and submit it to the District Coordinator (DC) for her approval. Otherwise other bodies of the bank (e.g., the Credit Office director -CO; the Resolution Body-RB) have to write the credit proposal and/or decide its approval. If the credit proposal gets the corresponding approval, the client receives a copy of the contract for signing it.

Figure 2a and Figure 2b present, respectively, the Workflow Net Model (WNM) and the Workflow Sequential Model (WSM) of the Credit Procedure described above.

The WNM (Figure 2a) makes explicit the local conditions of the process, so that the independence, for example, between the actions of 'Perfecting Practice' and 'Checking Financial Insolvencies' is easily visible. On the contrary, the WSM (Figure 2b) makes explicit the global states of the workflow, so that the path followed during the execution of an instance is made immediately clear.

Exception Handling

In designing its Credit Procedure (Figure 2) the Bank was not requested to specify the cases of withdraw or rejection. Even if they are frequent and relevant, they have been considered as special cases, to be treated as exceptions (since they truncate the workflow). Exceptions of any type provoke in the workflow either forward- (when some activities are omitted towards a final state) or backward- (when redoing some activities is required) jumps.

We can assume that our Bank allows two different classes of jumps: *strongly linear jumps* (moving only one token in the WNM) not requiring any type of authorization and *weakly linear jumps* (canceling two or more tokens and writing one token in the WNM) requiring the authorization of the Credit Office director.
Let us imagine, for example, that when an instance of the Credit Procedure is in the state \{b4, b9\} (see Figure 3a) the client withdraws; then the process instance can be brought to its conclusion through one of the allowed weakly linear jumps (all of them are represented by dashed lines in Figure 3b) moving the instance to the final state \{b19\}.

Let us also consider a different case: if in the same state one of the two active actors has a breakdown, so that she can not perform her activity, then she can either choose one of the strongly linear jumps (see the dashed lines in Figure 4) allowing her to redo or skip part of the investigation on the client, or to negotiate with the AD one of the weakly linear jumps of Figure 3b imposing either a restart of the process instance, or to jump forward skipping some of the required checks.

The negotiation between the actor who has a breakdown and the other actors and/or managers to decide which forward or backward jump is suitable in that case, is supported by Milano through its Conversation Handler (MCH). When the actor cannot perform the activity she has to do, then she can open a conversation with the other actors of the work process, where the module computing exceptional paths of any type can be enacted and executed. The above integration between MCH and MWMS is one of the most interesting characteristics of Milano (Agostini et al., 1997), since it supports the interlacing of ad hoc alignment with predefined procedures (Schmidt, Simone, 1996).

**Designing Changes**

Frequent recurrent exceptions may also require that the workflow model is changed in order to
minimize their impact on the future instances. If the Credit Office director, for example, analyzing all the Credit Procedure instances of the last three months, discovers that most of rejection cases were due to the discovery of some 'Financial Insolvencies', then she may decide that the analysis of the Financial Insolvencies should be performed as soon as possible in order to avoid unnecessary activities as 'Checking external Credits' in case of client's insolvency. Therefore, she decides to change the workflow model accordingly -see Figure 5.b- modifying the order of execution of the two activities (i.e., swapping them).

What the Credit Office director (or the Workflow Designer working for her) needs is being sure that the designed change is correct, i.e., that it does not create neither deadlocks nor undesired flows. While many approaches to workflow change correctness support only deadlock-freeness, as (van der Aalst, 1998), Milano evaluates the correctness of a change with respect to a Minimal Critical Specification (MCS) criterion, that can be also defined by default as the sequential automaton allowing any flow containing all the activities, so that it grants both its correspondence with the expectations and its deadlock-freeness. The correctness check is made through an injective morphism from the changed workflow model to the MCS (Agostini, De Michelis, 1998).

It may happen, that in order to increase immediately the efficiency of the Credit Procedure, the Credit Office director wants that the new model is enacted also in all the already ongoing instances (dynamic changes). In this case it is critical to know if an instance can continue correctly on the new model, or not.

To face this problem, Milano distinguishes in the old model, through a composition of morphisms, between safe and unsafe states. In short, one workflow instance is in a safe state if the latter has an image in the new model; moreover, the instances in safe states can continue correctly on the new model. Let us underline that we consider incorrect even those cases in which, moving to the new model, some activities should be ri-executed. In Figure 5a the unsafe states are shadowed.

The class of changes introduced above is still quite small, since it does not allow any change at the activity level. An extension of the allowed changes may be obtained weakening the condition that all the workflow models correct with respect to a MCS contain the whole set of activities occurring in the latter and/or extending the morphism definition to refinements.

Conclusion

The specification module of the Milano Workflow Management System can be considered as a demonstrative example of our theory-based approach to workflow model adaptability. Its originality lies in avoiding to look for specific techniques for supporting adaptability, while developing a theoretical framework where the standard flow is considered as the generator of views, exceptions...
and changes, that can be computed when necessary. Our example is based on a quite small class of
net systems, having various interesting properties. It would be quite useful if other classes of system
models are investigated from the same point of view.

Acknowledgments

Milano is under development with the financial support of European Community (within the Euro-
Canadian cooperation project on Cooperative Information Systems) and of the Italian Ministry of
University and Research. Special thanks are due to our students, Roberto Tisi, Paolo Bertona, Pietro
Nardella and Mario Manzoli, who greatly contributed to the development of the workflow
management system of Milano.

References

ACM Press, New York, NY, pp. 113-120.


Proceedings of the Fifth European Conference on Computer Supported Cooperative Work, Kluwer Academic,

Models, Techniques and Tools, Computing Science Report 98/07, Eindhoven University of Technology,
Eindhoven, pp. 146-164.

A Persistent Object Approach, Submitted for Publication.

Cooperative Work on the Print Industry Shopfloor. In Proceedings of the Fourth European Conference on
pp. 51-66.


De Michelis, G. and Grasso, M. A. (1994): Situating conversations within the language/action perspective: the

of the Conference on Organizational Computing Systems, Milpitas, CA, August 13-16. ACM Press, New York,
NY, pp. 10-21.

Science, 96,1, pp. 3-33.


Schmidt, K. (1997): Of maps and scripts: the status of formal constructs in cooperative work. In GROUP'97,


