The COO operator to support and improve the flexibility of adaptive workflows

Olivier Perrin, Claude Godart

The *coo* operator to support and improve the flexibility of adaptive workflows

O. Perrin and C. Godart
LORIA-INRIA
Campus Scientifique, BP 239,
54506 Vandœuvre-lès-Nancy Cedex – FRANCE
email: {Olivier.Perrin, Claude.Godart}@loria.fr

Abstract— **During the past few years, a lot of work has been done in the process modeling field. Process modeling is one of the major activity in domains like software engineering, design or creation. Researches focused on how to express models and how to ensure that a given model is well followed by all the participants. Despite these proposals, there is a few work on how to ensure that a model is suitable for an organization and that the proposed model is really usable. With new technologies getting important and dedicated to the World Wide Web, people realizes that some of the models are not fitting with the reality. Cooperation and collaboration aspects are often missing, and the models are not enough flexible to reflect the reality. All these problems are currently an important research field in two domains: software engineering processes ([1]) and workflow management systems([2]). The need for adaptive and flexible workflows is now well accepted, as the need for a higher level of cooperation. We present in this paper an approach, which uses techniques described in papers like [3], [4], [5] and original techniques based on the *coo* operator ([6], [7]). We also explain how this work is applied in a real AEC (Architectural, Engineering, Construction) application context and the reason why we choose such a solution. Then, we discuss our current work of implementation.**

I. Introduction

During the past few years, there is a significant amount of research done in modeling and supporting process models. In the software engineering domain, a lot of papers and experiences are proposed, and with the growing of business process reengineering, the models used to describe and support processes tend to rely on workflow model, as introduced in [8]. The workflow management is the automated coordination, control, and communication of work, both of people and computers, in the context of organizational processes, through the execution of software in a network of computers whose order of execution is controlled by a computerized representation of the business processes ([9]). These concepts are well defined, and there is now an important set of tools (called Workflow Management Systems) insuring that all the activities of a process are performed in the right order, and that the execution is done respecting the constraints of the workflow model.

In the project we are currently involved in, we found the task dedicated to execution not so hard to achieve, but the task dedicated to model the process a little bit more tricky. We are working with architects and the first problem we encountered concerns the acquisition of the initial workflow model. Because of the human factors, we faced a lot of difficulties to clearly express a model that satisfies the needs of architects. Then, we also faced problems due to the need of a higher degree of flexibility because of the ways architects cooperate. In fact, we understand that architects work in this way: they start from coarse grain workflows from which they derive real workflows with a lot of diversity, and that they are not able to model *a priori*. In this context, we face a challenge that contains two major issues:

- the first one is dedicated to the ability to discover and to tune a process learning from the usage, rather than from classical meetings and interviews,
- the second is to provide a high degree of flexibility, meaning that users should be able to modify and adapt the process with respect to human aspects.

We present in this paper our approach, which is based on two concepts. The first one is a tool that is able to infer a process model from history logs. It uses techniques that are described in [3], and uses the hidden markov models method. Then, once the model is capture, we use an operator - the *coo* operator - which offers more flexibility than those describe in the Workflow Management Coalition reference model ([8]). This operator also ensures the consistency and the integrity of data items.

The rest of the paper is organized as follows. In

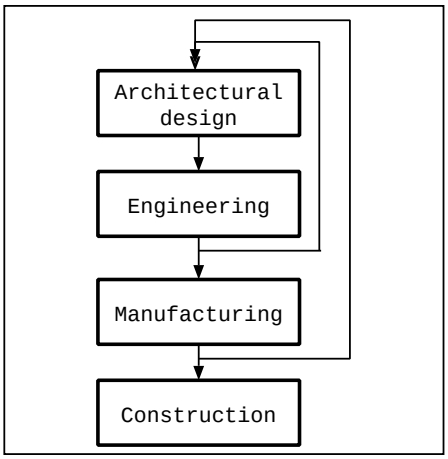

Fig. 1. Buiding trade example: rigid modeling

section 2, we present our case study, the problems we encountered and the requirements we had to fulfill. In section 3, we describe our proposal which combines two concepts, hidden markov models and the *coo* operator. Then, we conclude in section 4 by summarizing the benefits and the constraints of our proposal, indicating the ways we are currently looking for.

II. Our case study

Let us explain how we proceed with our project partners and how we discovered that some of the constraints expressed by the architects are not really applicable with current techniques. The context of this work is more widely described in [10].

In a first experience, we had to develop a simplified process dedicated to the construction of a building. This process involves three partners: an architect, a civil engineer and a building contractor. Activities allocated to each participant are defined as follows. The architect is responsible for producing the plans responding to the requirements given by the client. This is modeled as an activity in our system. Based on the previous plans and on its own expertise, the engineer makes the technological choices (choice of wood technology, where to put doors, gates, ...). Then, the building contractor is responsible to manufacture and to assemble the components on the site according to the choices made by the engineer and the plans made by the architect.

With current systems and the operators found in [8], we can model this process as an interactive process, such as depicted in the figure 1. At the end of each activity, we give the ability to return to the previous activity.

When we presented this model to our partners, the overall comment we receive from them is that this process, as being modeled, does not reflect reality for two main reasons. The first one is the simplicity of the process, in the sense that these activities are far more complex than just producing plans, making technological choices and manufacturing. Since it was only the first iteration of the model, we agreed on this comment, and the next iteration of the model was much more precise. Due to lack of space, we do not provide the new schema here. Briefly summarized, the complete schema encloses four phases: the feasibility phase (identification of needs, feasibility studies, final site selection), the design phase (schematic design, design development, construction documents, pre-bid, accept building), the construction phase (estimate project, construction, partial inspections, final inspection, payement) and the operation phase (final acceptance, building management). Each of these phases encloses itself sub-phases that are not described here.

The second reason was more interesting as it was directly related to the way we modeled the process. Architects reported that real processes do not execute the way we depicted it. Processes corresponding to the three partners are more intricated and seem to execute in parallel rather than isolation. They exchange documents when executing with the objective to put emphasis on a rapid feedback to point out risks and exceptions as soon as possible. In one scenario of our case study, the building contractor asked the engineer to move an opening due to transportation constraints (due to size of wood wall components). This decision had a lot of impacts especially with regards to the client wishes. Introduction of a model as the one we proposed can delay this evidence and contribute to poor acceptance of the workflow model. Figure 2 shows the new model obtained using the *coo* operator. Then, an other comment raised the fact that even if we try to make a very precise model, it will not be able to define all the situations that could lead to problems, and so, the model will not reflect the reality of their work, even the more precise it will be. They gave us the example of an appeal made by people during the building phase and concerning the area where the trade was supposed to be. In this case, they had to stop and to return to the design phase, keeping some existing studies and making new ones. Another example was the bankruptcy of one of the contractor. In this case, the building had not to be stopped: the ability to deviate from the process model must be provided. Summarizing these comments was an important contribution for us, and it was clear that current workflow systems are not suitable for our partners.

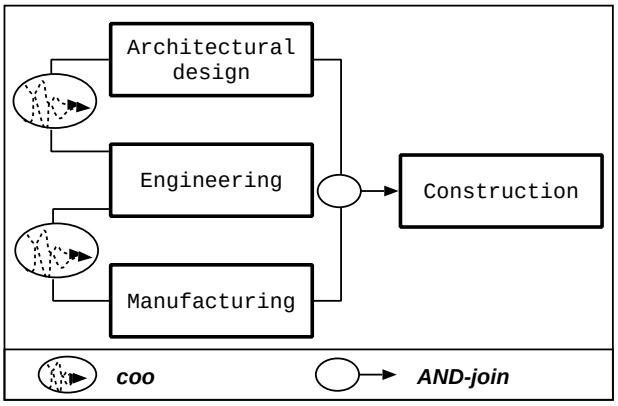

Fig. 2. Buiding trade example revisited : flexible modeling

We learned a number of very useful lessons, and we decided to set three major requirements that our work must take into account. The first requirement is to provide a more dynamic modeling phase for users. This means that we should be able to model the process before its execution, as usual, but users should have the ability to modify and/or to adapt the model they currently use in order to be able to deviate from or to dynamically extend the workflow model during the process execution, i.e. at run-time. The second requirement is dedicated to provide a higher degree of flexibility to reflect the human aspects of collaboration (synchronous/asynchronous work, formal/informal nature, deviations, backtracks, exceptions,...). As before, this need of flexibility should be available during both the modeling and the execution phases. This means that we should have operators that could reflect the reality of the cooperation and that we should provide new operators that are more flexible than the AND-split, AND-join, OR-split and OR-join operators. The third requirement is that we should be able to guarantee the correctness and the consistency properties of the new operators.

III. Our proposal: using HMM and *coo*

A. Hidden Markov Models

In order to provide an adaptive model, we took into account a study made in [1]. The authors studied three methods to discover models from event-based data. Their conclusion is that the Markovian method is better in many cases. In [11], the method used relies on DAG, and the process model was infer using FlowMark logs. Then, the prototype described in [5] uses HMM – hidden markov model – in order to achieve the acquisition of a workflow.

It appears that the markovian method provides several advantages against the following characteristics that are ([1]) :

- accuracy : it is guaranteed to generate an accurate model for the data it is given if their noise thresholds are set to zero.
- speed : this method is relatively quick, and another feature of this method is its independance from the length of the event stream (the size of data).
- tunability and usability : it is only tunable in a few dimension but using it is straightforward.
- robustness to noise : the method has the potential of being good at inferring models in presence of collection of errors and abnormal process events.

In our proposal, we use the classical HMM algorithm proposed by Rabiner ([3]) augmented with some of the features described in [4]. This approach is very similar to the approach described in [5]. This approach permits some benefits that are :

– evolution,
– flexibility, by dynamic recomputation of the model,
– deviation: unplanned events, rollback, forward recovery...

As said before, during the phase dedicated to the creation of the model, we had a lot of difficulties to get information on how architects really work, how they communicate, how they handle exception situations, how they capture evolutions in the model, and several situations they claimed not to be aware of a priori. In fact, it was difficult for them to precisely describe all the possible activity sequences in advance. For us, the results of this first phase was disappointing, so we have to develop a new approach that will allow us to capture their model, even if this model is not full featured. We found the HMM concept very attractive, and it fits very well with our needs. The first results we obtain was very satisfying.

In our approach, people using the tool have to provide a basic model, and the list of known activities. Then, the user can create new activities, including temporal constraints of their executions. By computation of these logs (activities and temporal constraints) with our tool, we extended and refined the model. We focussed on the modeling of behavioral aspects, rather than on the modeling of relationships between artifacts or the modeling of roles and responsibilities of agents. In fact, at the begining, we modeled only few activities, and then, with traces from executions, we were able to refine the model according to usage, new activities introduction or new contractor cooperations for instance. Of course, all these steps required that the participants manually told to the system which activity they completed once they finished it. However, we find it as a reasonable balance.

We can trigger the recalculation of the model either manually or automatically. In fact, we are able to :

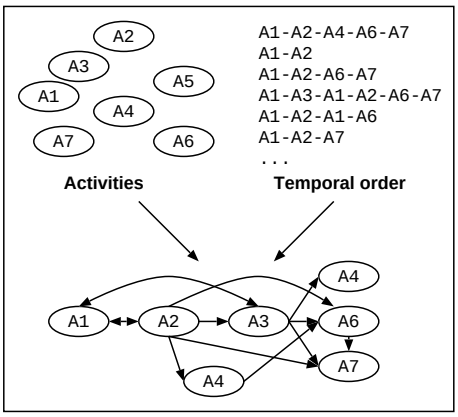

Fig. 3. Infering from logs

- compute the difference between the logs and the current model,
- recompute the model with respect to the logs.

The following figure shows how with a set of activities (the set is extensible with new activities) and the temporal constraints of execution, we can infere a process model. This model can then be enacted. The second interest is that, once enacted, traces are always kept and that we can adjust the model at any time with new activities or with new temporal constraints. This allows for deviation. Once the user enter within a deviation, its deviation is stored, and it is always possible for the user to go backward (and then to go forward again). Another feature is given to the user to put the trace mode active or inactive. This means that the user can explore a way without being traced by the tool. In this case, there is no information about this path, and no information can be retrieved and can be used to refine the model.

As the processes described by the architects are impractical to be entirely specified, we provided the ability to dynamically refine the specification of the model at runtime. This gives the ability to define or to definitely specify all the activities, and this also gives the ability to define interdependencies between these activities. To achieve this goal, we consider that the user can adjust both activities and interdependencies.

To conclude, we believe that this proposition represents a flexible and adaptive way of modelling activities, and that the use of the *coo* operator constitutes a very interesting approach in order to improve the modelling and to support the flexibility provided while guaranteeing some properties we will now introduce.

B. The coo operator

As introduced before, we also provide flexibility with a new operator that we are now going to describe. Once it seems that the model is mature enough and that it reflects the main parts of the requirements expressed by the architects, we use this model and we introduce the *coo*-operator everywhere we need cooperation and collaboration between activities and/or people. In fact, when the previous step is done, it is easy to see what are the activities where people need cooperation. In our example, we could easily see that the activities A1 and A2 are not necessarily in one order or another. This means that these two activities can be modeled at the same level. Classical operators from the WfMC reference model are too rigid to model this kind of situation. That is why we propose a new operator, the *coo* operator. We previously said that we are only concerned with temporal constraints of activities in the first step. With the *coo* operator, we are able to take into account the other aspects of cooperation (such as modeling relationship between artifacts).

The *coo* operator gives the opportunity to simplify the modeling and the enactment of cooperative workflows. It does not cover all the aspects of cooperation, but it is well adapted when processes interact *asynchronously* during their execution by exchanging several successive *versions* of artifacts. Exchanges can be uni-directional or bi-directional. We believe that *coo* is fine in three major styles of cooperation:

- *producer/consumer.* Two activities follow a *producer/consumer* pattern if one has the responsability to modify an object and the other reads this object to use it in its own work. The producer and the consumer can momentarily see two different versions of the object, but they must be agree on the final version of the shared object before the producer stabilizes it.
- *developer/inspector.* This pattern defines the case where one activity (the developer) produces an object and another (the inspector) reads one or many successive versions of this object with the commitment to review it.
- *cooperative write.* Two activities follow a *cooperative write* pattern if they develop two complementary versions of the same object and they must merge their respective modifications before to conclude. They have the ability to render visible some intermediate versions of their work before finishing.

The syntax of the operator is $coo(a_1, a_2)$, where a_1 and a_2 define two activities. The operator collaboration with other operator is easy since *coo* does not set any restriction on the structure of the activities it synchronizes.

The semantics of the *coo* is defined by the *coo*-serializability correctness criterion. We quickly summarize the rules defining the protocol to use it.

1. the result produced before the end of an activity

is always an intermediate result. A user can produce an intermediate result at any time using the *IR-write* operation.
2. a result produced at the end of an activity is a final result (produced during the execution of the *terminate* operation).
3. an activity that produces an intermediate result must produce a corresponding final result.
4. if an activity reads an intermediate result, then it must read the corresponding final result (the system maintains *dependency relationships* between activities to memorize the fact that an activity reads an intermediate result of another one).
5. an activity cannot terminate if it still dependent of another one. If an activity tries to terminate without reading the final value of an intermediate value, the *terminate* operation is aborted and the activity remains active.
6. activities involved in a cyclic dependency graph form a *group* of activities.
7. a group-member activity can start a *group termination* by trying to terminate itself. In this situation, *terminate* produces a set of potentially final results and changes the state of the activity from *active* to *ready to terminate (RTT)*.
8. when a group-member activity tries to terminate and when all the other group-members are in the *RTT* state, then all the activities terminate simultaneously.
9. when a group-member activity tries to terminate and if there is another group-member still active, then it produces new potentially final results and enters the *RTT* state.
10. if a group-member produces a new intermediate result during the group termination phase, then this termination tentative is aborted, and all the group-members re-enter the active state. This is a way for an activity to clearly indicate a disagreement with the object values produced by the group, and to ask revision on the shared object.

Then, it is possible to configure the *coo* operator thanks to two parameters. The first one is called *direction* and it can take three values (<-->, <--, -->) to indicate either if the interaction is bidirectional or the direction of the exchange. The second parameter, called *mode*, can take two values (*all* or *atdemand*) and indicates if all intermediate results must be consumed or not.

IV. Conclusion

The results we currently have with the architects are very hopeful. They found that the method used to model their processes is really what they are waiting for. It provides a dynamic refinement capability at run-time, a concept very important for people that cannot completely and definitely specify some of their activities. Moreover, the flexibility of the approach is also a major point for the acceptance of such tools. As their processes are often revised, they have now a powerful tool that is able to follow their needs and the way they work. The fact that unpredictable events can be integrated by dynamically refining the process or changing the current activity is also appreciated. Then, human aspects of collaborative processes are took into account thanks to the *coo* operator.

Acknowledgments

We wish to thank O. Malcurat and J.C. Bignon for their fruitful discussions, suggestions and comments all along this work, and France Telecom for supporting this project.

References

[1] J. Cook and A. Wolf, "Discovering Models of Software Processes from Event-Based Data," *ACM Transactions on Software Engineering and Methodology*, 1998.
[2] Y. Han, A. Sheth, and C. Bussler, "A Taxonomy of Adaptive Workflow Management," in *Proceedings of the ACM Conference on CSCW – workshop Towards Adaptive Workflow Systems*, 1998.
[3] L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE*, 1989.
[4] A. Stolcke and S. Omohundro, "Best-First Model Merging for Hidden Markov Model Induction," Tech. Rep., International Computer Science Institute (ICSI), 1994.
[5] J. Herbst and D. Karagiannis, "Integrating Machine Learning and Workflow Management to Support Acquisition and Adaptation of Workflow Models," in *Proceedings of the Ninth IEEE Conference on Database and Expert Systems Applications*, 1998.
[6] C. Godart, G. Canals, F. Charoy, P. Molli, and H. Skaf, "Designing and Implementing *COO*: Design Process, Architectural Style, Lessons Learned," in *International Conference on Software Engineering (ICSE18)*, 1996, IEEE Press.
[7] C. Godart, O. Perrin, and H. Skaf, "*coo*: a workflow operator to improve cooperation modeling in virtual processes," in *Proceedings of the Ninth International Workshop on Research Issues on Data Engineering RIDE-VE '99*, 1999.
[8] WfMC, "Terminology & Glossary," Tech. Rep., Workflow Management Coalition, 1997.
[9] S. Joosten, G. Aussems, M. Duitshof, R. Huffmeijer, and E. Mulder, "WA-12: An Empirical Study about the Practice of Workflow Management," Tech. Rep., University of Twente, 1994.
[10] J.C. Bignon, G. Halin, K. Benali, and C. Godart, "Cooperation models in co-design: application to architectural design," in *4 th International Conference on Design and Decision Support Systems in Architecture and Urban planning*, Maastrich, July 1998.
[11] R. Agrawal, D. Gunopulos, and F. Leymann, "Mining Process Models from Wokflow Logs," in *Proceedings of the Sixth International Conference on Extending Database Technology (EDBT)*, 1998.