Using Fuzzy Petri Nets to Coordinate Collaborative Activities

Alberto B. Raposo, André L. V. Coelho, Léo P. Magalhães, Ivan L. M. Ricarte
DCA, FEEC, State University of Campinas (Unicamp)
Campinas, SP, Brazil
{alberto, coelho, leopini, ricarte}@dca.fee.unicamp.br

Abstract
This paper presents a fuzzy Petri net based approach suitable for the modeling of flexible coordination mechanisms to deal with temporal interdependencies between collaborative tasks. Such approach is based on an extension of the Generalized Fuzzy Petri Net model, including the notion of time for the execution and synchronization of these tasks. A scenario of study is described, indicating the suitability of the proposal.

1. Introduction
A collaborative activity (CA) is defined as a coordinated set of tasks realized by multiple actors to achieve a common goal. Tasks are the building blocks of activities. They can be atomic or composed of subtasks, and are connected to one another through dependencies.

The coordination of computer-supported CAs, which is the act of managing interdependencies between the tasks, is a very important and difficult endeavor [1]. The great challenge in proposing coordination mechanisms to control a CA is to achieve the flexibility demanded by the dynamism of the interaction between the partners [2].

We have defined a set of interdependencies that frequently occur in CAs and proposed coordination mechanisms (modeled using Petri nets - PNs) for these dependencies [3]. The idea is to separate tasks from dependencies (controlled by the coordination mechanisms), enabling the use of different coordination policies in the same collaborative domain, by changing only the coordination mechanisms. Moreover, these mechanisms may be reused in other collaborative domains.

Nevertheless, it is sometimes very difficult to completely define the interdependencies underlying CAs. This happens because those relationships may already embed in their essence a not so well defined (or fuzzy) semantics. Such modeling imprecision implies, as a positive side effect, more flexibility to application designers as they can focus on their own definition of the relation, in a manner more closely related to the subjective human reasoning. Using the resources offered by the fuzzy sets theory, interdependencies in CAs can assume a more manageable perspective, improving both the understandability and the feasibility of the interacting rules that identify the whole process.

This paper presents a fuzzy PN-based approach suitable for the modeling of flexible coordination mechanisms for CAs. In the following, some aspects related to CAs and their inner interdependencies, as well as the use of PN as a coordination tool, are briefly pointed out. In Section 3, we describe the fuzzy PN approach and its simulation algorithm. A case study comes in Section 4. The paper resumes with conclusions and suggestions of future research.

2. Coordination of Collaborative Activities
It is possible to characterize different kinds of interdependencies that frequently occur in CAs and identify coordination mechanisms to manage them [1]. Among them, we can highlight temporal interdependencies, which encompass those situations in which it is necessary to establish an execution order for tasks, resembling a scheduling process.

To create the coordination mechanisms for those dependencies, we have employed an approach based on PNs [3]. The graphical representation of PNs, besides being clear, allows detail encapsulation, offering an adequate infrastructure to model different coordination levels. Furthermore, PNs offer a strong support for analysis and simulation.

Taking into account temporal interdependencies, the employment of conventional PNs brought with itself some restrictions on the modeling of the aforementioned coordination mechanisms, since tasks can only be synchronized by their start or finish times. This rigidity limits the analysis of the activities of a range of common collaborative scenarios, since it prohibits, for instance, commencing a second task...
when the first is "almost finished". In this sense, this paper presents a further step towards the development of flexible temporal coordination mechanisms for CAs by means of fuzzy PNs, extending results achieved in a former work [3].

3. The Fuzzy Petri Net Model

There are several approaches that combine fuzzy sets and Petri nets theories, differing not only in the fuzzy tools used, but also in the elements of the nets that are fuzzified (tokens, markings, or transition firings). Amongst them, we choose the Generalized Fuzzy Petri Net (GFN) [4] as our basis because of its consistence with the Boolean PN model (i.e., a safe PN, where places can have only 1 or 0 tokens) and its powerful and flexible way to fuzzify not only the transition firing rules, but also the tokens flow through the net.

3.1. GFN – Generalized Fuzzy Petri Net

The GFN set of state update equations is straightforward and very malleable to compute, since it is based only on simple inferences over t- and s-norm aggregations. The degree of a transition firing is given as [4]

\[ z = \bigcap_{i=1}^{n} \left( t_{i} \rightarrow x_{i} \right) s w_{i} \]  

where \( T \) and \( s \) denote, respectively, a t- and an s-norm, \( x_{i} \in [0,1] \) is the (fuzzy) marking of input place \( i \) of the transition (it has a total of \( n \) input places), and \( w_{i} \) is the weight factor that represents the contribution of place \( i \) to the overall level of firing. The symbol \( \rightarrow \) denotes a fuzzy implication defined as

\[ a \rightarrow b = \sup \{ c \in [0,1] \mid a \ t \ c \leq b \} \]  

where \( t \) is a t-norm. Therefore, \( r_{i} \) in Equation (1) modulates the "strength of firing coming from the \( i \)-th input place" [4].

A transition in a GFN is enabled if its degree of firing \( z \) is equal or higher than a threshold level \( \lambda \in [0,1] \). Once an enabled transition is fired, each of its input places will have its marking decreased according to the following distribution

\[ x_{i} (\text{next}) = x_{i} \ t \ z' \]  

where \( z' \) is the complement of \( z \).

Similarly, each output place \( j \) of the transition will increase its marking \( y_{j} \) according to Equation (4)

\[ y_{j} (\text{next}) = y_{j} \ s \ z \]  

As in conventional PNs, it is also possible to define inhibitory input places for a transition (Equation (5)). The only difference between the excitatory term \( z \) and the inhibitory term \( z' \) is that the latter is a function of the complement of the input place’s marking \( (x_{i}') \).

\[ z = z^{+} t z^{-} = \bigcap_{i=1}^{n} \left( t_{i} \rightarrow x_{i} \right) s w_{i} \bigcap_{j \neq i}^{n} \left( t_{j} \rightarrow x_{j} \right) s w_{j} \]  

The GFN model, however, was developed for rule-based systems (actually, it is based on neural nets) and does not address the timing of transition firings. For our synchronization purposes, it is crucial to include the notion of time in the model. For that reason, we have developed an extension of the GFN model presented in the following.

3.2. Extending GFN

Our extension of the GFN model includes the notion of time by means of two distinct and independent time intervals that can be associated to each transition: the firing delay \( d = [t_{\min}, t_{\max}] \) and the firing duration \( \Delta \). The former delimits the minimum and maximum time between the enabling of the transition and its firing [5]. The value of \( t_{\min} \) establishes the minimal time that must elapse between the time the transition becomes enabled and the time it can fire. The value of \( t_{\max} \) defines the maximal time the transition can be enabled without firing. At this time, the transition must fire if it has not fired yet (and if it is still enabled). Using this model, it is correct to state that, if a transition fires, this happens in a time \( t_{f} \in d \), where \( t_{f} \) is measured relatively to the time the transition becomes enabled.

Due to its delay, the transition may not be fired immediately once enabled (the tokens remain in their input places), being possible that other transitions remove them, disabling the delayed transition. In this case, it will not fire in \( t_{\max} \).

The delay introduces a non-fuzzy temporal uncertainty to the firing of the transitions. There is a couple of approaches that fuzzifies the firing delay of transitions (e.g., [6], [7]), but we use the original non-fuzzy delay [5] in our current implementation.
The second time interval associated to each transition in our GFN extension is the firing duration $\Delta$ [8]. The value of $\Delta$ defines the “execution time” of a transition. If $\Delta > 0$, its firing is not an instantaneous (atomic) operation. This kind of non-instantaneous firing is important to encapsulate details of the tasks execution, allowing us to represent these tasks (which are not instantaneous) simply as transitions in our model.

Originally, the value of $\Delta$ associates a deterministic firing duration and two events (start firing and end firing) to a transition [8]. The start firing event removes tokens from the transition’s input places. The end firing event ($\Delta$ units of time later) adds tokens to the transition’s output places. In between these events, the firing is in progress.

A consequence of the use of non-instantaneous firings is that the state of the net cannot be defined only by its token distribution. It is also necessary to verify the firings which are in progress and their remaining firing times (RFTs).

In order to give more flexibility to the specification of temporal interdependencies between tasks, the transitions that represent these tasks have their firing duration $\Delta$ fuzzified. In this case, $\Delta$ is seen as a linguistic variable defined over a temporal scale. The designer is the one who stipulates the amount, the meaning, and the format of the membership functions associated to the linguistic terms ranging over each $\Delta$ partition (so, each transition can be customized). The linguistic terms allow for the definition of time parcels representing possible synchronization points between two tasks. As illustrated in Figure 1, the duration $\Delta$ is associated with three linguistic terms (beginning, middle, finish), whose membership functions overlap. In this case, $P_1$ and $P_2$ are two synchronization points given by

$$
\begin{align*}
P_1 &= \inf_{t \in \Delta} \text{middle}_\tau \\
P_2 &= \inf_{t \in \Delta} \text{finish}_\tau
\end{align*}
$$

(6)

where

$$
\begin{align*}
\text{middle}_\tau &= \{ t | \text{middle}(t) \geq \tau_1 \} \\
\text{finish}_\tau &= \{ t | \text{finish}(t) \geq \tau_2 \}
\end{align*}
$$

(7)

are the $\alpha$-cuts of middle and finish in relation to the universe of discourse defined by $\Delta$ and to the threshold levels defined by $\tau_1$ and $\tau_2$, respectively.

In order to directly specify the temporal interdependence between a pair of tasks, say $<T_1, T_2>$, the designer delimits a threshold value explicitly indicating the minimal degree to be achieved by the linguistic term (synchronization point) for assuming that “$T_1$ is fired in relation to $T_2$”. This strategy allows for the “relative firing” between different pairs of tasks. Therefore, $<T_1, T_2>$ (e.g., $T_2$ will commence when $T_1$ is “almost finished”) and $<T_1, T_3>$ (e.g., $T_3$ will commence after the “very beginning” of $T_1$) should incur distinct synchronization points for the same transition $T_i$. In the PN model, this means that the transition representing $T_i$ will release the tokens to an input place of $T_3$ shortly after its beginning and to an input place of $T_2$ only later, shortly before its end. It is worthy to note that this abstraction does not imply that the tasks themselves have fuzzy temporal periods associated to them, but simply that the tokens in the model of the coordination mechanism will be released in accordance with the fuzzy semantics of their temporal interdependencies.

It is important to reinforce that $d$ and $\Delta$, although independent, have complementary roles, in the sense that they may be applied simultaneously in the same transition, but the use of one does not imply the use of the other. The delay is employed to insert a non-deterministic behavior in the modeling of tasks execution. Conversely, the firing duration introduces fuzzy synchronization points by means of which the tasks can have their execution interrelated.

3.3. Simulation Algorithm

In order to model CAs using the proposed fuzzy PN-based approach, we have developed an algorithm to
simulate the behavior of such PNs. The algorithm reads an input file that describes the structure of the net (places and their initial markings, transitions and arcs) and its additional parameters: \( r_i, w_i, d, \Delta \) and \( \lambda \) for each transition, and the membership functions (beginning, middle and finish) and synchronization points (given by a dominant function and a threshold value \( \tau \), such as in Eqs. (6) and (7)) for each output place of each transition. The output of the simulation is a file describing, at each instant, the state of the PN, the firing rule \( z \) for each transition, the list of enabled transitions, and the next state change. Figure 2 shows the pseudo code of this simulation algorithm.

4. Applying the Model to Collaborative Activities

In the proposed coordination model, CAs can be described in two abstraction levels [3]. In the workflow level, each actor’s activity is modeled separately by a PN, in which his/her tasks are represented by transitions. Also in this level, it is necessary to establish the interdependencies between the tasks. The coordination level, on the other hand, is built under the workflow level by the expansion of interdependent tasks according to a model defined by W. van der Aalst [9] and the insertion of corresponding coordination mechanisms between them. Coordination mechanisms, in this context, are reusable pre-defined PNs which ensure that the associated interdependency between tasks will not be violated.

By means of the proposed GFN extension, we are able to redefine the whole set of coordination mechanisms for temporal interdependencies underlying CAs in a more flexible way. This set of temporal interdependencies was proposed elsewhere [3] and is based on the interval logic of J. Allen [10], which was adapted to the context of CAs.

Amongst temporal interdependencies, we choose the \textit{equals} relation as our case study. This relation establishes that two tasks must be executed simultaneously. In a hypothetical manufacturing system, for example, the arms and the legs that will compose a doll must be produced at the same time by different machines. Therefore, we can say that both tasks (production of arms and production of legs) have the \textit{equals} interdependency. Formally speaking, considering two tasks \( T_1 \) and \( T_2 \) that occurs, respectively, in intervals \([t_1_l, t_1_f]\) and \([t_2_l, t_2_f]\), then \( T_1 \text{ equals } T_2 \) iff \( t_1_l = t_2_l \) and \( t_1_f = t_2_f \).

\begin{verbatim}
do while not end
    Select_firing
    if time == max_simulation_time then
        end = true
    for each transition in progress Tp
        for each output place of Tp
            if synchronization point then
                send tokens to output place
                exit Select_firing
            if \( \Delta(Tp) \) is finished then
                finish execution
        for each disabled transition in its delay time
            reset delay
        for each enabled transition Te
            if Te is not yet in its delay time then
                set delay
            if it is time to fire Te (i.e., \( time = \text{random value } \in d \)) then
                select Te
            if there is a selected transition then
                go to Fire
            else if there is no transition in progress AND no enabled transition before its delay interval then
                end=true
                exit Select_firing
    Fire
    if there is a selected transition Ts then
        for each input place of Ts
            remove tokens from input place
        if \( \Delta(Ts) > 0 \) then start execution of Ts
        if \( \Delta(Ts) = 0 \) then
            for each output place of Ts
                send tokens to output place
            exit Select_firing
    End Fire
    increment time
End Select_firing
End do while
\end{verbatim}

Figure 2: Simulation algorithm for the GFN extension.

The coordination mechanism for this interdependency is shown in Figure 3. Places start\textunderscore task, indicates that task, is ready to begin and finish\textunderscore task, indicates it is finished. Transitions \( t_1 \) and \( t_2 \) ensure, respectively, the simultaneous beginning and end of both tasks. The transitions called task, are non-instantaneous (\( \Delta > 0 \)) and represent the logistic of each task.
Either Boolean or fuzzy constructs could be applied in this coordination mechanism. The great limitation of this coordination mechanism using Boolean PN is that each task can start only when both are ready and finish only when the other also finishes. This rigidity can cause frequent deadlocks in the PN, especially if the tasks belong to complex CAs. For example, if the actor that is supposed to realize task2 has an alternative path that does not execute this task, the other task (task1) can be blocked. In Boolean coordination mechanisms, this problem is reduced by the use of timeouts, which execute alternative tasks if the original ones have not been executed after a certain waiting time. Using fuzzy coordination mechanisms, the number of deadlock situations is reduced because a task not completely ready can enable the execution of another.

Another aspect on the use of fuzzy coordination mechanisms is that the degree of parallelism in the execution of tasks is enhanced. In the manufacturing system example, the production of legs could be started a bit before the production of arms, because it takes a little longer (i.e., the first task may start when the other is almost ready). To “quantify” this situation we use the product as t-norm (t b = ab), the probabilistic sum as s-norm (s b = a + b - ab) and consider that transition t1 has r1 = 0.8 and w1 = 0.5 for both input places. If the first task is completely ready (x_start_task1 = 1) and the other is 70% ready (x_start_task2 = 0.7), then, according to Equation (1), z(t1) = 0.9375. Therefore, if the threshold λ(t1) is equal or less than this value, then t1 will fire (even if task2 is not completely ready) sending tokens to places P1 and P2, which can enable the execution of task1 and/or task2. A similar situation can occur in relation to transition t2 at the end of the tasks.

The notion of synchronization points can be used to commence a third task before the actual end of task1 and task2. In the manufacturing system example, this third task could be the assembly of the doll, which needs the arms and legs resulting from task1 and task2. In the Boolean situation, this task may start only after the end of the first two tasks. However, if there is an initial phase of this task that does not require these pieces (e.g., to fit the head in the body of the doll), it could be started when the pieces are, say, 80% finished. For this situation, a synchronization point

$$P = \inf_{t \in \Delta} 0.8$$

can be defined for task1 in relation to its output place P5. Considering that the membership functions of the linguistic terms for Δ are three gaussian functions $G(x) = \exp\left[-0.18(x-center)^2\right]$ centered, respectively, on 0, Δ/2 and Δ (similar to Figure 1), then $P = \Delta - 1.11$, which means that a token will be sent to P5 1.11 units of time before the end of task1. A similar approach may be used for task2.

The comparison with the Boolean case can be made using several criteria, such as time elapsed until a given state is reached, number of states needed to reach a desired state, number of deadlock situations, among others. For the example presented in this section, we can make a straightforward comparison considering the time needed to achieve the end of both tasks (tokens in finish_taski). In the fuzzy case, we initially may have a gain because it is not necessary to wait until the slower task becomes totally ready. Taking the values discussed previously, the waiting time to start task1 and task2 will be $\max\{t_{r_{task1}}, 0.7*t_{r_{task2}}\}$, where $t_{r_{taski}}$ is the time needed for taski to become ready. In the Boolean case, this waiting time will be $\max\{t_{r_{task1}}, t_{r_{task2}}\}$. If $t_{r_{task2}} > t_{r_{task1}}$, there will be a gain. The other gain in the fuzzy case is related to synchronization points in the execution of the tasks. Once tokens may be released before the actual end of the tasks, the desired state can be reached earlier. Therefore, we can conclude that, in the fuzzy case, the desired state is reached in a time which is less or (in the worst situation) equal than that of the Boolean case.

Although rough, the above comparison clearly indicates that using fuzzy coordination mechanisms designers are able to manipulate tasks interdependencies according to a very flexible semantics and create CAs with an improved degree of parallelism.
5. Conclusion
This paper presented two distinct contributions. The first one is the extension of the GFN model to include the notion of time. The second contribution is the employment of GFN-based coordination mechanisms in CAs, which provides a higher degree of flexibility in the definition of tasks interdependencies, when compared to the approach using conventional PNs. This is particularly noticeable for the reduction in the use of timeouts, which are necessary in the conventional approach to avoid the frequent deadlocks caused by the rigidity of coordination mechanisms [3].

The case study also showed that the use of fuzzy coordination mechanisms may improve the degree of parallelism in the execution of CAs and give them a more manageable perspective. The price to be paid for this flexibility is that designers must deal with a larger number of variables. However, by means of simulation tools, they can have powerful support to lighten this difficulty.

The next step of this research will be the complete formalization of the GFN extension and the definition of qualitative and quantitative metrics to prove the efficiency of fuzzy coordination mechanisms. We also intend to extend the use of fuzzy coordination mechanisms to other kinds of interdependencies besides temporal ones (e.g., resource management dependencies [3]).

Finally, we reinforce that task coordination is a problem that should be addressed to ensure the effectiveness of CAs. The use of fuzzy coordination mechanisms can bring to this field a higher degree of flexibility that should be explored.

6. Acknowledgement
The first and second authors are sponsored by FAPESP (00/10247-3) and CNPq (140719/1999-7), respectively.

7. References