Anticipatory Alignment Mechanisms for Behavioral Learning in Multi Agent Systems

Gerben G. Meyer, Nick B. Szirbik

Department of Business & ICT, Faculty of Management and Organization, University of Groningen, Landleven 5, P.O. Box 800, 9700 AV Groningen, The Netherlands, +31 50 363 {7194 / 8125}
{g.g.meyer,n.b.szirbik}@rug.nl

Abstract. In this paper we present a conceptualization and a formalization to define agents' behaviors (as exhibited in agent to agent interactions), via an extension of Petri Nets, and show how behaviors of different agents can be aligned. We explain why these agents can be considered anticipatory, and the link between Business Information Systems and anticipatory systems is elaborated. We show that alignment is a state anticipatory mechanism, where predictions about future states directly influence current behavioral decision making. This results in faster and more reliable interaction execution. Also, alignment provides a mechanism for more direct behavioral learning. We investigated three manners of alignment, individual on-the-fly alignment, pre-interaction alignment, and alignment with the intervention of a third party. This paper explains in some detail how alignment on-the-fly is realized using alignment policies. The features of the other two kinds of alignment are discussed, and future directions for research are pointed out.

1 Introduction

In the anticipatory system research community, the agent based computing area is considered a promising one. However, there is yet little interest in applying the anticipatory agent concept in a real setting. Seminal work of Davidsson, Astor and Ekdahl [5], pointed out that active entities can be characterized as agents when their acting can be described by a social theory. We argue in this paper that business organizations are in fact anticipatory systems themselves. Especially when these use an information system (usually called BIS - Business Information System). Our research group is investigating novel agent-based architectures and development frameworks [13]. We recognize the importance of the anticipatory system concept in this context and position our models of organizations in the initial definition of Rosen ([14], page 339):

> "We tentatively defined the concept of an anticipatory system: a system containing a predictive model of itself and/or of its environment, which allows it to change state at an instant in accord with the models prediction to a latter instant."

In this paper, we investigate how the anticipatory ability of a single agent can be expressed as an interaction belief and we point out how in some cases this belief can be changed. We describe a policy for alignment that can be applied when the interaction beliefs of two or more interacting agents are not matching. We introduce an extension of Petri Nets to capture the interaction beliefs and also a mechanism to choose the appropriate policy that adapts the beliefs from one agent perspective. Furthermore, we discuss the case when the process of alignment before the actual interaction takes place. Also, alignment by a third-party is investigated. From the anticipatory systems perspective, this research can enable predictive agent model execution (agent-based simulation of organizational models) to be more reliable and necessitate less human intervention in terms of alignment.

1.1 Motivation

Business information systems have evolved from a data centric perspective to a process centric perspective. The role of these systems is to support human activity in a business organization. At a basic level they support information storage and retrieval, information flow and information processing. At a higher level they support human decision making. Depending on the time horizon, the decision can be related to operational management (day-to-day activities), tactical planning (week/month projections), strategic decisions (month/year projections), and even policy implementation (very long term).

The move from data centric to process centric systems did not change the centralistic nature of these systems. The way the system is designed and used ascribes to the notion that there exists an external observer that is able to investigate and understand the processes within the organization. These processes can be identified in a semantic sense and modeled in a syntactic sense, that is, models of the processes can be described in a (semi) formal language. These models can be used to implement systems that support the actors who execute the process in the organization.

The actors who are executing the organizations' processes have only local, often conflicting views, especially in dynamic organizations. If the system is to be designed and implemented by allowing local and different models of the participating actors, a distributed, agent-oriented approach is more suitable. Agent-based modeling and agent-software engineering have been very popular in the last decade and have paved new avenues for the development of the business systems of tomorrow. However, due to the lack of a strict definition of an agent and a clear view about what exactly agent software engineering is, as pointed out by Ekdahl [6], many developmental processes tend to be termed agent-oriented, although they really can be just classified as purely reactive systems. Ekdahl also states:

> "More sophisticated anticipatory systems are those which also contain its own model, are able to change model and to maintain several models, which imply that such systems are able to make hypotheses and also that they can comprehend what is good and bad."

One can infer from this statement that true agent systems are only those that have a clear anticipatory ability, both at the level of the individual agents themselves, and also at the whole multi-agent system level. The ability to reason about a plan in an organization is usually realized via humans. If one tries to simulate a planning organization, a typical barrier is the evaluation of the plans. Such simulations tend to become interactive games, where the "players" (i.e. the expert planners) are becoming decision makers that select the "best" plan. Various plan selection mechanisms can be enacted, but these are usually just models of the behavior of the players. In a monolithic, centralistic system for example, this will be implemented as a single utility function that characterizes the whole organization, which makes explicit the criteria against which a prospective plan is checked. In reality, many expert players are co-operating with the system to adjust and decide for the best plan. The overall behavior of the organization (in terms of planning) is just emerging as a combined behavior of the experts and the system that supports them.

This observation leads to the natural conclusion that it is better to enact decision support structures that mimic the distributed nature of this environment. Attempts to model and implement agent-oriented support for planning and other business processes are still in their infancy, but even simple implementations of crude multi-agent architectures show a higher degree of adaptiveness and flexibility.

We envisage the first wave of applying these kind of agents to repetitive and routine business processes, like the ones in sales and purchasing, financial operations, and operational control of logistic and manufacturing systems [11]. Here, the need for anticipatory based alignment is rather low, but it is easy to execute in an automated way, based on typical policies that have been detected over time. One can say that this application area can be seen as robotics, but in this case the robots are not physical entities, but digital entities (sometimes these are called "softbots").

1.2 Our Approach Towards Anticipatory Agents

Our research team is developing agents via simulation-games, where the behavior of the software agents is deduced from expert players. These human experts can describe their intended behavior, in terms of activities and local goals, but also can describe the behavior they expect from the other agents in the game. These behaviors can be simplified and formally described. From a local perspective the *intended behavior* of self and the *expected behavior* of others can be seen as a specific *interaction belief* of that agent. The organizations' processes can be viewed as a set of running interactions. Each interaction is executed by the agents that play the roles that define the interaction and the execution depends on the (local) interaction beliefs. If the agents have beliefs that are consistent with each other, a coherent execution of the interaction will take place. In an environment where human agents are playing the roles, slight (or even severe) misalignment of these behaviors can be solved by the capacity of the humans to adapt to misunderstandings and information mismatch.

Agents (as humans) develop over time a large base of interaction beliefs, which allow them to cope with a wide range of interaction situations. This is why the organizational processes can be carried out in most contexts and exceptional situations. When using an agent-oriented approach, in order to solve the exceptions that occur but have no resolution beliefs implemented in the software agents, an "escape/intervention" [13] mechanism can be used. Each time an agent cannot find a local solution for a mismatch during an interaction, it can defer control to a higher authority (higher level agent, typically a human). Therefore, such a system will never block, supporting the humans up to the levels it has been programmed to do, but leaving the humans to intervene when the situation is too complex for them to solve.

Interaction beliefs are local anticipatory models. These describe future possible states in a specific interaction from a local perspective of an agent. In an organization, an agent can play various roles by using its "experience" (interaction beliefs that have proved successful in the past), but can also build new ones, depending on the context. Continuous enactment of interaction leads to whole process enactment. In a software multi-agent system, if the captured behaviors are not matching in a given context, the agents will revert to humans. Of course, this can decrease the performance of the system - in terms of support and/or automation - to unacceptable levels. Software agents should also be able to overcome their mismatching behaviors in an anticipatory way. There are multiple ways to tackle behavior mismatches:

- Each agent is individually trying to align its behavior on-the-fly, having only local information.
- The agents try to align their behavior before the interaction starts, by sending and comparing each other's intended behavior.
- There is a third-party agent interfering with the interaction:
    - The third-party agent can be a superior agent that can align and impose a common interaction behavior that is sound, by having full access to the interaction beliefs of the agents. This can happen before the interaction starts.
    - The third-party agent acts as a mediator between the agents.

A typical interaction where such behavioral mismatches can occur, is the buyer-seller interaction, which will be elaborated on in the later parts of the paper. The most encountered behavioral mismatch in the buyer-seller interaction is due to the fact that each party wants to have its output criteria fulfilled first. Basically, that means that the buyer wants to be in possession of the product before he pays, and the seller wants to receive the payment before delivery. In this example, a typical third-party agent interfering with the interaction is a bank, who takes the risk of paying the seller first, and invoicing the buyer after he received the product. Later in the paper, we show how these behavioral mismatches can be solved using various alignment mechanisms.

1.3 Taxonomy and Benefits

In the on-the-fly mechanism, the anticipatory system is the individual agent who tries to align its behavior, based on the limited information it has about the interaction execution. According to the taxonomy of Butz et al. [4], this mechanism is a state anticipatory mechanism, as predictions about future states directly influence current behavioral decision making. In this case, a predictive model must be available to the agent, or it must be learned by the agent. In our approach, such a model is formalized using Behavior Nets, as will be explained in the next section. The Behavior Net captures the planned behavior of an agent for the interaction it intends. The on-the-fly alignment mechanism, as proposed in this paper, will result in faster and more direct model and behavioral learning, as the agent is able to learn new behaviors during the interaction. Furthermore, it will result in improved social skills, as the agent is able to alter its behavior during the interaction, in order to ensure a successful interaction, even when the original behaviors of the interacting agents are not matching.

When aligning behavior before the actual interaction (or pre-interaction alignment), future states do not directly influence current behavioral decision making, instead future states expected by the agents are used to discuss their course of actions, in order to align their beliefs about the interaction. For this reason, it can still be called state anticipation, as the explicit predictions about the future (formalized as Behavior Nets) influence the discussion process, and thus the future behavioral decision making, which is defined in the Behavior Nets of the agents. This form of anticipatory behavior will be beneficial for social behavior within the overall system, as predictive knowledge of other agents is exploited.

We considered that the third choice with a superior agent is “less anticipatory”, in the sense that only if viewed from a larger perspective (the system is formed by the participating agents, plus a third-party agent), it becomes a system that investigates a potential scenario for the future. The mediator on the other hand exploits the predictive knowledge of the interacting agents, for aligning them through its own behavior. For this reason, this approach will also be beneficial for social behavior.

1.4 Paper Outline

In the next section, we introduce a representation of the behavior in terms of Behavior Nets. In section 3, we describe a mechanism for individual alignment based on "alignment policies". Section 4 describes the pre-interaction alignment, and section 5 the alignment with a third-party agent interfering. We end with a discussion and conclusions.

2 Behavior Nets

Petri Nets are a class of modeling tools, which originate from the work of Petri [12]. Petri Nets have a well defined mathematical foundation, but also an easily understandable graphical notation [15]. Because of the graphical notation, Petri Nets are powerful design tools, which can be used for communication between the people who are engaged in the design process. On the other hand, because of the mathematical foundation, mathematical models of the behavior of the system can be set up. The mathematical formalism also allows *validation* of the Petri Net by various analysis techniques.

The classical Petri Net is a bipartite graph, with two kind of nodes, *places* and *transitions*, and directed connections between these nodes called *arcs*. A connection between two nodes of the same type is not allowed. A transition is *enabled*, if every input place contains at least one token. An enabled transition may fire, which will change the current *marking* of the Petri Net into a new marking. Firing a transition will *consume* one token from each of its input places, and *produce* one token in each of its output places.

2.1 Definition of Behavior Nets

In the following, the formal definition of Behavior Nets is given, which is a Petri Net extension, based on Workflow Nets [1], Self-Adaptive Recovery Nets [8] and Colored Petri Nets [9]. An example of such a Behavior Net can be seen in figure 2 (a).

Definition 1. *Definition of Behavior Nets*

A Behavior Net is a tuple $BN = (\Sigma, P, Pm, T, Fi, Fo, i, o, L, D, G, B)$ where:

- Σ is a set of data types, also called color sets
- P is a finite set of places
- Pm is a finite set of message places
- T is a finite set of transitions (such that $P \cap Pm = P \cap T = Pm \cap T = \emptyset$)
- $Fi \subseteq ((P \cup Pm) \times T)$ is a finite set of directed incoming arcs, and
- $Fo \subseteq (T \times (P \cup Pm))$ is a finite set of directed outgoing arcs, such that:

$$\forall p \in Pm : \bullet p = \emptyset \oplus p\bullet = \emptyset$$

- i is the input place of the behavior with $\bullet i = \emptyset$ and $i \in P$
- o is the output place of the behavior with $o\bullet = \emptyset$ and $o \in P$
- $L : (P \cup Pm \cup T) \to A$ is the labeling function where A is a set of labels
- $D : Pm \to \Sigma$ denotes which data type the message place may contain
- G is a guard function which is defined from Fi into expressions which must evaluate to a boolean value
- B is a binding function defined from T into a set of bindings b, which binds values (or colors) to the variables of the tokens

The set of types Σ defines the data types tokens can be, and which can be used in guard and binding functions. A data type can be arbitrarily complex, it can for example be a string, an integer, a list of integers, or combinations of variable types.

The places P and Pm and the transitions T are the nodes of the Behavior Net. All three of these sets should be finite. The extension of classical Petri Nets is the addition of the set Pm which are nodes for sending and receiving messages during an interaction. Such a message place is either a place for receiving or for sending messages, it cannot be both.

Fi and Fo are the sets of directed arcs, connecting the nodes with each other. An arc can only be from a place to a transition, or from a transition to a place. By requiring the sets of arcs to be finite, technical problems are avoided, such as the possibility of having a infinite number of arcs between two nodes.

Executing a behavior is part of an interaction process, the behavior is created when the interaction starts, and deleted when the interaction is completed. For this reason, the Behavior Net also has to have one input and one output node, because the Behavior Net initially has one token in the input place when the interaction starts, and can be deleted when there is a token in the output place.

With function L, a label can be assigned to every node. This has no mathematical of formal purpose, but makes the Behavior Net more easily understandable in the graphical representation.

Function D denotes which message place may contain what data type. This is useful for determining which message place an incoming message has to be placed on. Because the two (or more) behaviors in an interaction are distributively executed, message places of both behaviors cannot be connected directly with each other, as the behaviors do not have to be aligned.

Function G is the guard function, which expresses what the content of a token has to be, to let the transition consume the token from the place. Function G is only defined for Fi, because it makes no sense to put constraints on outgoing edges of transitions. In other words, this function defines the preconditions of the transitions.

Transitions can change the content of a token. Binding function B defines per transition, what the content of the tokens produced by the transition will be. Bindings are often written as, for example, $(T1, < x = p, i = 2 >)$, which means that transition $T1$ will bind value p to x and value 2 to i. The values assigned to the variables of the token (which data type must be in Σ) can be constants, but can also be values of the incoming token, or values from the knowledge- or belief-base of the agent.

2.2 Operations

In Behavior Nets, there is a set of *primitive* operations for modifying the net structure, such as adding and deleting places, transitions, arcs and tokens. Besides the primitive operations, there is a set of more *advanced* operations, which also preserve local soundness. By preserving local soundness we mean that after applying the operation, an execution of the behavior will still terminate properly, if the behavior is also terminated properly before the operation. The message places Pm are not taken into account when determining local soundness. Local soundness refers to a sound behavior, to make the distinction with a sound interaction, which will be referred to as global soundness. More information about soundness can be found in [2]. The used set of advanced operations are:

- `division` and `aggregation`*, which divides one transition into two sequential transitions, and vice versa,
- `parallelization` and `sequentialization`*, which puts two sequential transitions in parallel, and vice versa,
- `specialization` and `generalization`, which divides one transition into two mutual exclusive specializations, and vice versa,
- `iteration` and `noIteration`, which replaces a transition with an iteration over a transition, and vice versa,
- `receiveMessage` and `notReceiveMessage`, which adds or deletes an incoming message place,
- `sendMessage` and `notSendMessage`, which adds or deletes an outgoing message place.

For some of the operations, marked with *, is it not always clear how they can be applied on-the-fly, because of the dynamic change problem [3]. For example, `sequentialization`, as mentioned above, cannot be applied for every token marking, as it is not always clear on which places the tokens from the old behavior should be placed, when migrating to the new behavior. For modeling the migrations the approach of Ellis et al. [7] is used. By modeling a behavior change as a Petri net, it can be exactly defined how to migrate the tokens from the old behavior to the new behavior. Note that advanced operations can also be described using the primitive operations. For the `receiveMessage`, `notReceiveMessage`, `sendMessage` and `notSendMessage`, nothing needs to be migrated, as there is no change in the places, except for the message place, which initially do not contain a token. In figure 1 can be seen how the migration for the operation `parallelization` can be modeled.

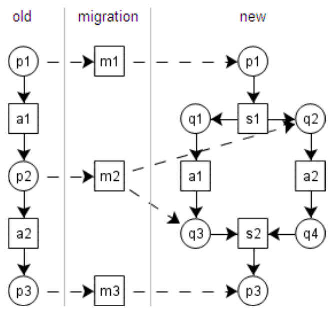


Fig. 1: Migration of old to new behavior

3 Individually Aligning Behaviors On-the-fly

Before two agents start an interaction, they will both individually choose a behavior they are going to execute, based on what they are expecting out of the interaction. Such behavior is an explicit representation (a Behavior Net) of the course of actions (transitions) and future states (places) the agent is expecting. An interaction however will not terminate, if the planned behaviors of the agents interacting are not matching. When the behaviors are not matching, certain expected future states are unreachable. Therefore, the agent has to alter its predictions about the future. To do so, agents must be able to change their behavior on-the-fly, i.e. during the interaction. For this purpose, alignment policies are used by the agents.

3.1 Alignment Policies

An alignment policy is an ordered set of primitive or advanced operations. In our approach, an agent has a set of policies in his knowledge-base from which it can choose when an interaction for example has deadlocked, i.e. when there is no progression anymore in the execution of the behavior. How an agent will choose an alignment policy (or if it will choose one at all) depends on different factors. The factors used are: problem information, beliefs about the agent being interacted with, and the willingness to change its own behavior.

Problem information. Most of the time, a problem will occur, when the agent is not receiving the message it is expecting. It can also be the case that the agent did not receive a message at all. If it did receive a message, the type of the received message and other information about the problem can be used as attributes for selecting the proper alignment policy.

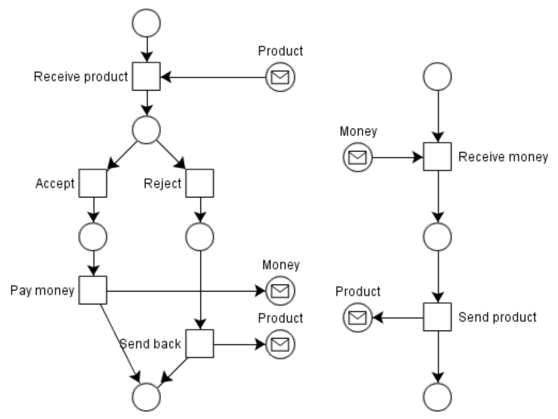
Beliefs about the agent being interacted with. Beliefs about the other agents can be of great importance when choosing an alignment policy. For example, when the agent completely trusts the other agents, it might be willing to make more change in its behavior than when it distrusts the other agents.

Willingness to change behavior. When an agent has very advanced and fine-tuned behaviors, it is not smart to radically change the behaviors because of one exceptional interaction. On the other hand, when the behavior of the agent is still very primitive, changing it a lot could be a good thing to do. Hence when an agent gets "older", and the behaviors are based on more experience, the willingness to change its behavior will decrease. This approach can be compared with the way humans learn, or with the decrease of the learning rate over time when training a neural network.

Currently, we are experimenting with agents who use a Neural Network to choose an alignment policy [10]. As this approach seems to be promising, other approaches, like Genetic Algorithms, could also be used. When an agent does not know how to handle a certain problem, it can go into escape mode, to learn new ways to overcome its lack of experience. With a neural network, you exactly know the certainty of the agent choosing a specific alignment policy. In this way, it is easy to know when to trigger escape mode. More information about the concept of escape mode can be found in [13].

3.2 Example

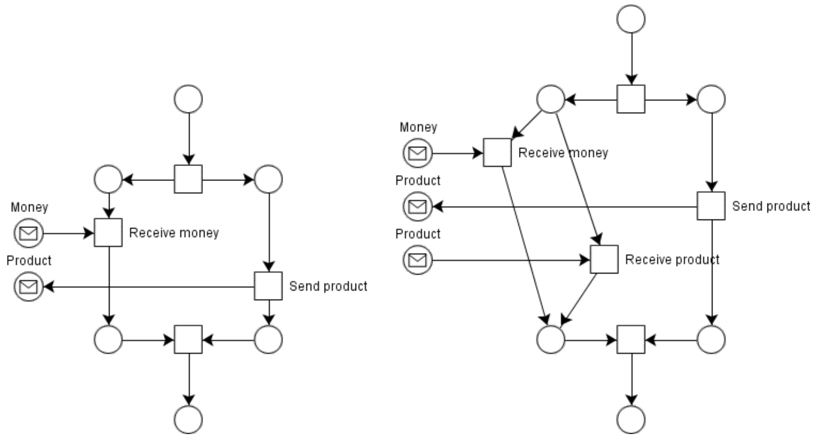
This section gives an example of how these alignment policies could work. In this example, as shown in figure 2, the buyer and the seller already agreed on the product the buyer wants to purchase, but, as seen in the figure, they have different ideas about the order of the delivery and the payment. For the sake of the example, we assume that the behavior of the buyer is very advanced, and thus has no willingness to change its behavior. On the other side, the seller's behavior is still primitive and unexperienced, hence we are looking at the problem of how the seller can align its behavior with the buyer, assuming that the seller has trust in the buyer.



(a) Buyer (b) Seller

Fig. 2: Behaviors of buyer and seller

When the interaction starts, it immediately deadlocks; the buyer is waiting for the product, and the seller is waiting for the money. A way to overcome this problem would be for the seller to send the product and wait for the money in parallel. Therefore, by using an alignment policy based on the operation `parallelization` the behavior of the seller changes to the behavior as seen in figure 3 (a), and the interaction can continue. However, if the buyer rejects the product, and sends it back, the seller still doesn't have the appropriate behavior to handle this, because the seller is waiting for the money. In case the seller receives the product back, but when it is expecting the money, the seller could use an alignment policy based on the operation `specialization` to overcome this problem, which divides the receive money transition into two separate transitions, *receive money* and *receive product*. The resulting behavior can be seen in figure 3 (b). The behaviors of the buyer (figure 2 (a)) and the behavior of the seller (figure 3 (b)) are now aligned, and thus matching.



(a) First adaptation (b) Second adaptation

Fig. 3: Adapted behavior of seller

4 Pre-interaction Alignment

As said before, the agents participating in an interaction will choose a priori a behavior they intend to execute. These behaviors consist of their own intended actions and future states. Conflicting behaviors within an interaction can be solved on-the-fly, as seen in the previous section, but another approach would be to align the behaviors before the actual interaction. In this way, the expected future states of the agents can be aligned beforehand. Such a pre-interaction alignment is a very difficult process. Although it is natural between humans, it is very difficult to formalize and implement a solution that offers pre-interaction alignment for software agents. In this section, we are discussing merely the complexity and problems poised by pre-interaction alignment. An example based on an operational business process illustrates the problem.

4.1 Pre-interaction Alignment With Intended Behaviors Only

There are two ways in which pre-interaction alignment can be achieved. In the first case, we assume that the agents posses only the description of their own intended behavior. Before an interaction starts, the agents that are committed to execute the interaction can be forced by the designer of the interaction-based system to reveal their intended interaction to each other, like in figure 4.

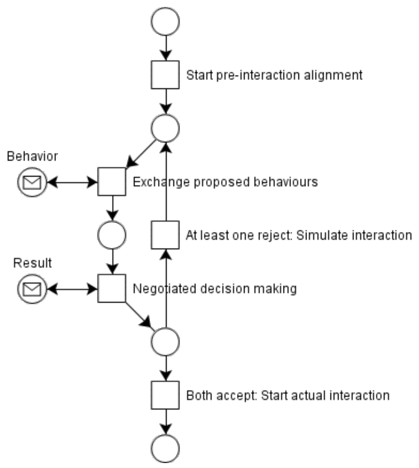

Start pre-interaction alignment
Behavior
Exchange proposed behaviours
At least one reject: Simulate interaction
Result
Negotiated decision making
Both accept: Start actual interaction

Fig. 4: Behavior for pre-interaction alignment

Each agent can automatically infer whether the interaction will succeed or fail. If both agents realize that their behaviors are not matching and they have internal interaction simulation abilities, each can run simulations of the future interaction and come out with behaviors that matches each other. These anticipatory matching behaviors can be exchanged again, and if the agents agree via negotiation about using one, they can start the actual interaction.

In an interaction with only two agents, there are four possible outcome scenarios of the agents' simulations:

1. Both agents propose new behaviors for each other.
2. One agent proposes a new behavior, and the other accepts it.
3. One agent proposes a new behavior, and the other rejects it.
4. No agent is able to propose a new behavior.

After scenario 1, a following round of negotiation and, if necessary, exchanging new behaviors is needed. In order to reduce the complexity of the pre-interaction negotiation about how to align the behaviors, some rules for selecting a matching behavior can be enacted. For example, only scenario 1 may be allowed, and the selection is based on an "external" assessment, i.e. the simplicity of the result (in terms of the number of places, activities and arcs). Or, alternatively, scenario 1 is applied in successive steps, and via an internal assessment, for example, the results are graded 1 to 10 by each agent, leading to the selection of the one with the highest cumulative grade. This implies of course that the agents should be able to assess independently, without human intervention, the quality of a behavior they have to exhibit.

4.2 Using Interaction Beliefs

Pre-interaction alignment can be improved if the agents also have beliefs about the expected behavior of the other one(s). We are currently extending the language to model behavior and behavioral belief by adding notation for expected behavior. Because this represents the expected course of action and future states of the agent being interacted with, exchange of intended and expected behaviors could more easily reveal potential mismatches in the expected future states of different agents. In this setting, one agent represents for itself the intended behavior for a given interaction, by using a normal Behavior Net, and also Behavior Nets that represent what this agent is expecting from the other agents involved in the interaction.

For example, consider the situation in a small enterprise where the sales manager has to expedite an order for a particular customer. He intends to send a request for re-planning the order to the planner and also a request to the shop-floor scheduler, who is in charge of the actual early re-scheduling of the tasks necessary to execute the order faster. In figure 5, the sales manager intends to send the requests, waits for an answer from the scheduler and depending on the answer (re-scheduling possible or not) continues to handle the negotiation with the customer. He expects that the scheduler will analyze the request, and if feasible, will arrange the re-scheduling and give a positive answer. He also expects that the planner will update the delivery date for the order in its own system.

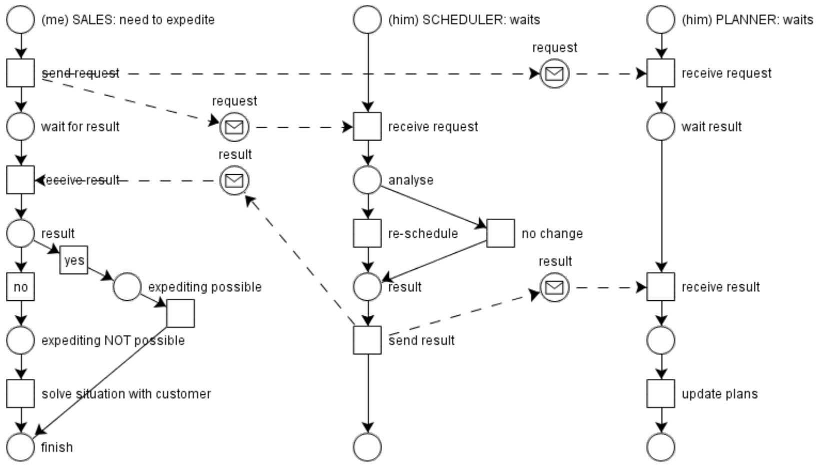


Fig. 5: Interaction Belief of sales manager

The scheduler (as in figure 6), is not cooperative in this interaction. When he receives the request, he is just making a justification for refusal and sends this back to the manager. He is not eager to make himself a new schedule, due to various reasons (e.g. lack of time). His beliefs about the sales manager reflect his unhelpful behavior, because he beliefs that the sales manager made a mistake that resulted in the order expedition and it is his role to solve the problem with the customers. He is not aware for example that expedition of orders can increase the price the customers pay. By behaving in this way, the sales manager will always have a negative response. We call the graphs depicted in figure 5 and figure 6 Interaction Beliefs (IBs), comprising the intended behavior of an agent and also the expected behavior of the others.

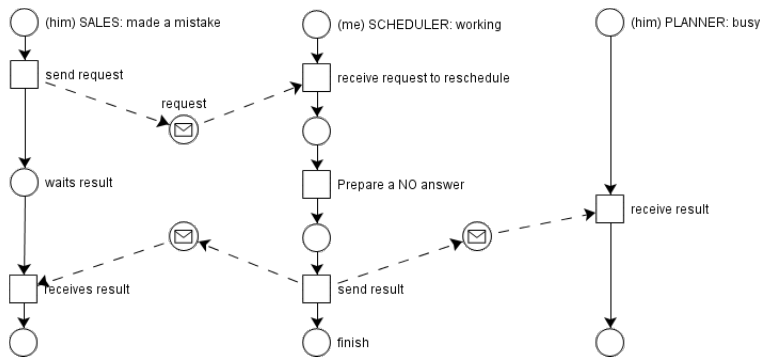
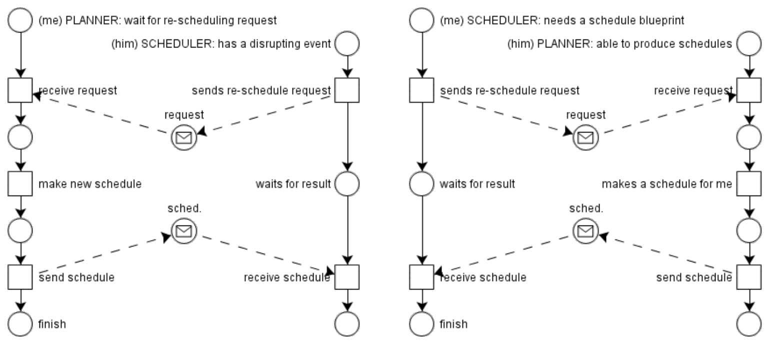


Fig. 6: Interaction Belief of scheduler

On the other hand, the scheduler is aware that the planner has an automatic scheduler used to check the feasibility of the automatically generated plans (capacity check). In figure 7, we depict two matching IBs, in an interaction between the scheduler and the planner. When the scheduler has a problem at the shop-floor level, and needs a new schedule, he asks the planner to provide him with one generated by the schedule-based capacity checker. The scheduler can use this as a blue print for a new schedule and will send the updated schedule to the planner, who will update the plans. However, the planner will not generate a schedule at the expedition request of the sales manager, but will only change the delivery date for the order.

If these three agents would exchange their IBs depicted above, the sales manager will immediately realize that the scheduler will always give him a negative response. Also, he will realize that there is a powerful scheduling capability with the planner, and the shop-floor scheduler tends to use this capability instead of making its own schedules. The planner will realize also that expedited orders are never realized in the shop-floor, and its own scheduling ability is used only when the scheduler asks for it.

We envisage an alignment via negotiation and re-design process of the IBs, leading to matching that can use all this existent information in the expected behaviors. Currently, such a situation is solved by humans, who can design a solution like in figure 8. Here, the request for expediting is sent directly to the


(me) PLANNER: wait for re-scheduling request
(him) SCHEDULER: has a disrupting event
receive request
sends re-schedule request
request
make new schedule
waits for result
sched.
send schedule
receive schedule
finish
(me) SCHEDULER: needs a schedule blueprint
(him) PLANNER: able to produce schedules
sends re-schedule request
receive request
request
waits for result
makes a schedule for me
sched.
receive schedule
send schedule
finish

(a) IB of planner (b) IB of scheduler

Fig. 7: Matching IBs between the scheduler and planner

planner, who generates a new prospective schedule for the shop-floor, which is adapted and if feasible is sent back to the planner, who has to update his plans, together with a "green-light" for the sales manager.

It is also possible that the graph in figure 8 is the IB of a new planner who arrived in an organization where the sales manager and the scheduler behave and believe as shown in figures 5 and 6. We can consider that in this situation we have a "knowledgeable" planner agent, and two other "stupid" agents, who are just wasting each other's time. The planner can show how the interaction should be carried out, and the other two agents can "learn" from the IB, which one has better experience. Thus, the expected behavior in IBs (at least from ones of knowledgeable agents), can be used to spread experience to agents who are beginners.

The process of alignment becomes more complex if there are more than two agents involved in the interaction. Also, if the interaction is long, aligning becomes combinatorially complex. For this reason, it is advisable to attempt to align behaviors which are short and are applied in bipolar interactions. Business processes are in fact complex interactions. In order to achieve business process coherence, a method to decompose into simple interaction, as described in [16], is necessary. Furthermore, the use of expected behaviors via IB interchange will increase the complexity of the alignment process compared with the interchange of intended behaviors only.

5 Alignment by a Third-party Agent

In the previous two kinds of alignment, the interacting agents have to solve the conflict of non-matching behaviors themselves. However, it could be the case that the interacting agents did not manage making an agreement. A logical next step would be to let a third 'neutral' agent intervene in the interaction. This can

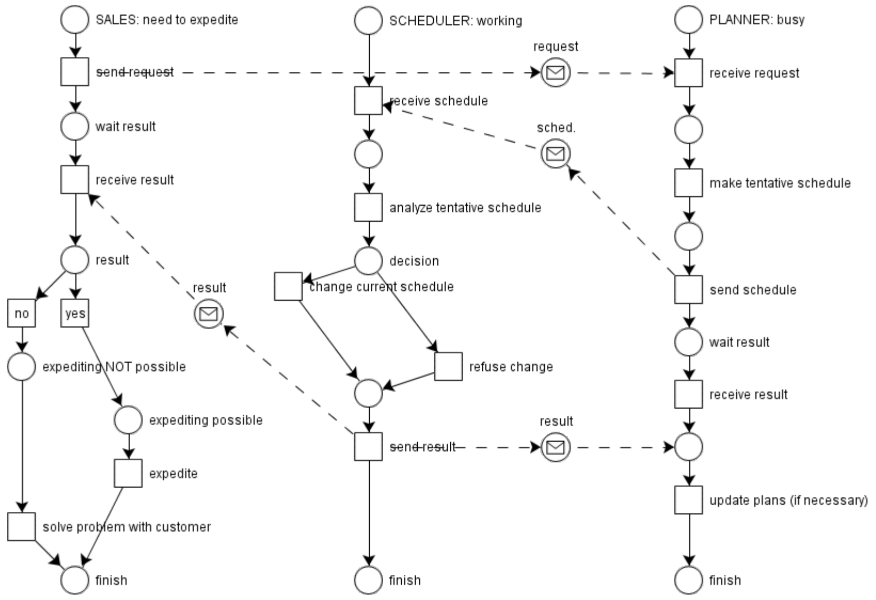


Fig. 8: Human solution

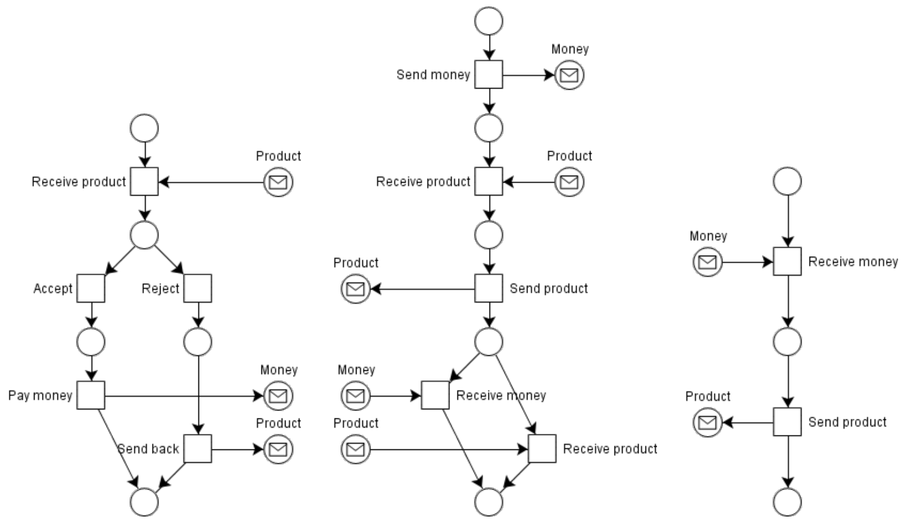
be done in two ways, the neutral agent being a mediator, or being a superior agent.

5.1 Third Agent as a Mediator

When there is a problem in the execution of a behavior, the agent can choose an alignment policy to solve this conflict, as described in section 3. Such an approach only works if one of the agents is willing to make "sacrifices" to its behavior (e.g. if the seller in the previous section is willing to send the product first, instead of first receiving the money). In such a case, an agent can also ask a mediator, by choosing a special policy. This can also be a policy learned from a superior agent, by the use of escape mode. For this purpose, it has to know an agent who is able to play the mediator-role for this particular interaction.

In the best case scenario, both agents can use their initial non-matching behaviors when interacting with the mediator. For this reason, when one agent calls for a mediator, there is no reason for the other agent in the same interaction to refuse this. But there is still no guarantee that the interaction will be performed successfully. The use of a mediator could require both agents to change their behavior, for example when the mediator wants to have a fee [17]. Individual alignment can be used to alter the behaviors of the initial interacting agents, to make them aligned with the mediator. This "sacrifice", however, can be refused by the agents.

An example of how an interaction between two agents with conflicting behaviors can be solved with a mediator is shown in figure 9. The same behaviors of the buyer and the seller as in the example of section 3 are used. However, for alignment in this case, these agents do not have to change their behavior, because they are only interacting with the mediator. The behavior of the buyer is matching with the behavior of the mediator, and the same is true for the seller. The mediator in this example (which can be a bank) first pays the product, passes the product through to the buyer, and earns the money from the buyer. This is the best-case scenario for this interaction, as the mediator does not want a fee. This is not likely in real life, but this is only an illustrative example.



(a) Buyer (b) The mediator (c) Seller

Fig. 9: Example of alignment with a mediator

5.2 Third Agent as a Superior Agent

When the interacting agents both have the same superior agent, this agent can be asked to solve the conflict. Again, this can be done by the use of a special policy, as with asking for a mediator. A superior agent can also be asked when pre-interaction alignment has failed. However, in this case, the third agent will not play a role in the interaction. The superior agent will compare the behaviors of both agents, and will "enforce" aligned behaviors to both agents. The superior agent can do this in the same way with simulation as described for the pre-interaction alignment in section 4, but without the negotiation, as the superior agent imposes the agents to use a specific behavior. This can be a way of escape mode itself, as the superior agent does not necessarily need to be a software agent. When the interacting software agents fail, a human can align the behaviors of the agents manually.

6 Discussion and Future Work

As far as we know, this is the first attempt to apply this kind of discrete mathematics to anticipatory agents. Although there are some approaches that apply Petri Nets to model agent interaction, these are mainly concerned with a centralistic view. However, we are taking a distributed approach. This approach has the potential to appeal to two research communities: the one oriented towards Business Information Systems development (who apply Petri Nets for modeling workflows), and also to the growing anticipatory agent community. Some researchers have pointed out that the models used for BIS analysis and design are in fact executable models of the organization they support. Apparently, the inclusion of a executable model of the organization in the organization itself (seen as a system), makes the whole an anticipatory system. Obviously, organizations that use a BIS increase their anticipatory ability. Unfortunately, there is no evidence that the current development of BISs is done with explicit anticipatory ability in mind.

Our strong belief is that agent-oriented BIS that support the business processes of the organization (in terms of interaction support), due to the anticipatory ability of the individual agents, lead to an emergent behavior of the whole system that has an anticipatory nature. Of course, such a statement has to be proven empirically and theoretically. An intuition is that simulation - currently intended for development purposes - can have an important role in the anticipatory architecture of an agent-enabled BIS in an organization. If the executable agent-based model of the organization can perform simulations itself that start from the present state as perceived in the organization, this model can predict future states. The results of these predictive simulations can be used to influence the current state of the organization via an effector sub-system.

Currently, the idea about development simulations is that these are in fact games, where expert players interact with the simulated agents, via the escape/intervention mechanism. An escape is triggered when an agent cannot perform a certain act, and an intervention is when the human supervisor decides that the course of action is not as desired. After the agents are fully developed and are deployed in the organization, the predictive simulations that they could perform should be as automatic as possible (otherwise human intervention would make this anticipatory mechanism inefficient). This observation proves the need for better automatic alignment mechanisms very relevant.

Our future research will be directed towards a number of issues. Mechanisms for triggering the escape mode should be investigated, but also what the human will do after the escape is activated, i.e. how to train the agents. Pre-interaction alignment also has to be further investigated. Mechanisms for simulation resulting in new behaviors, and negotiating what behavior to use, are in a preliminary stage. Furthermore, alignment with the intervention of a third-party agent needs further research as well.

7 Conclusion

As we have shown, it is possible to describe a policy for alignment that can be applied when the interaction beliefs of two or more interacting agents are not matching. We introduced an extension of Petri Nets to capture the interaction beliefs and also a mechanism to choose the appropriate policy that adapts the beliefs from one agent perspective. From the anticipatory systems perspective, this research can enable predictive agent model execution (agent-based simulation of organizational models) to be more reliable and necessitate less human intervention in terms of alignment.

As proposed in this paper, the on-the-fly alignment mechanism results in faster and more direct model and behavioral learning, as the agent is able to learn new behaviors during the interaction. Furthermore, it resulted in improved social skills, as the agent is able to alter its behavior during the interaction, in order to ensure a successful interaction, even when the original behaviors of the interacting agents were not matching.

As we have shown, pre-interaction alignment and alignment with the help of a mediator are beneficial for social behavior, as these approaches exploit predictive knowledge of other agents.

We consider a superior agent aligning the interacting agents forcing them into new "less anticipatory" behavior. For this reason, the other mentioned approaches are favored above this one. This approach can however be used as a last resort, especially because the superior agent can be a human.

Finally, we believe that interdisciplinary work between the BIS research and anticipatory agent research can yield lots of "cross-fertilization" and raise the awareness that BIS enabled organizations are in fact anticipatory systems and also provide test beds for novel anticipatory agent ideas. Today, there are many system architectures that support activities in business processes, like workflow systems and enterprise integrated systems. Most of these architectures are monolithic and centralistic, having a low degree of flexibility. An architecture based on agents that allow change via different mechanisms for behavioral alignment offers a more natural approach to process support.

We consider that business processes that are inherently interactional, like in sale/purchasing, but also planning and scheduling can benefit from these alignment mechanisms. There is a clear uptake in implementing adaptive sale/purchasing systems via web-services, but there is still low emphasis on higher-level issues related to these systems, like conceptual architectures, interaction representations and mechanisms. In contrast with technology-driven approaches, our approach is doing exactly that, taking a top-down, formalism based, research-oriented approach.

References

1. W.M.P. van der Aalst. Verification of Workflow Nets. *Lecture Notes in Computer Science*, vol. 1248, pp. 407-426, 1997.

2. W.M.P. van der Aalst. Interorganizational Workflows: An approach based on Message Sequence Charts and Petri Nets. *Systems analysis, modelling, simulation*, vol. 37(3), pp. 335-381, 1999.
3. W.M.P. van der Aalst and S. Jablonski. Dealing with workflow change: identification of issues and solutions. *Computer systems science and engineering*, vol. 5, pp. 267-276, 2000.
4. M.V. Butz, O. Sigaud and P. Gerard. Internal Modals and Anticipations in Adaptive Learning Systems. *Lecture notes in Artificial Intelligence*, vol. 2684, pp. 86-109, 2003.
5. P. Davidsson, E. Astor and B. Ekdahl. A Framework for Autonomous Agents Based on the Concept of Anticipatory Systems. In *Proc. of Cybernetics and Systems 1994*, pp. 1427-1431, World Scientific, 1994.
6. B. Ekdahl. Agents as Anticipatory Systems. In *Proc. of SCI'00 and ISAS'00*, pp. 133-137, 2000.
7. C.A. Ellis and K. Keddara. A Workflow Change is a Workflow. *Lecture notes in Computer Science*, vol. 1806, pp. 201-217, 2000.
8. R. Hamadi and B. Benatallah. Recovery Nets: Towards Self-Adaptive Workflow Systems. *Lecture notes in computer science*, vol. 3306, pp. 439-453, 2004.
9. K. Jensen. An Introduction to the Theoretical Aspects of Coloured Petri Nets. *Lecture Notes in Computer Science*, vol. 803, pp. 230-272, 1993.
10. G.G. Meyer. Behavior Alignment as a Mechanism for Interaction Belief Matching. *University of Groningen*, 2006.
11. G.G. Meyer. Intelligent Products: an Application of Agent-Based Ubiquitous Computing. Accepted for: *ABUC'07*, 2007.
12. C.A. Petri. Kommunikation mit Automaten. PhD thesis. *Institut fur instrumentelle Mathematik*, Bonn, 1962.
13. Gijs B. Roest and Nick B. Szirbik. Intervention and Escape Mode. In *Proc. of AOSE'06 workshop, at AAMAS'06 Conference*, pp. 109-120, 2006.
14. R. Rosen. Anticipatory Systems. New York: Pergamon, 1985.
15. K. Salimifard and M. Wright. Petri net-based modelling of workflow systems: An overview. *European journal of operational research*, vol. 134(3), pp. 664-678, 2001.
16. M. Stuit and N.B. Szirbik. Modelling and Executing Complex and Dynamic Business Processes by Reification of Agent Interactions. Accepted for *Lecture Notes in Artificial Intelligence (ed. M. O'Grady)*, 2007.
17. M. Stuit, N.B. Szirbik and C. de Snoo. Interaction Beliefs: a Way to Understand Emergent Organisational Behaviour. Accepted for: *ICEIS'07*, Funchal, Madeira, Portugal, 2007.