Towards Modeling and Simulating a Multi-party Negotiation Protocol with Colored Petri Nets

E. Bacarin[1], W.M.P van der Aalst[2], E. Madeira[3], and C. B. Medeiros[3]

[1] Department of Computer Science - UEL - CP 6001 86051-990 Londrina,PR Brazil. bacarin@uel.br
[2] Department of Mathematics and Computer Science, Eindhoven University of Technology, P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands. w.m.p.v.d.aalst@tue.nl
[3] Institute of Computing - UNICAMP - CP 6176 13081-970 Campinas,SP Brazil. {edmundo,cmbm}@ic.unicamp.br

Abstract. E-contracting, i.e., establishing and enacting electronic contracts, has become important because of technological advances (e.g., the availability of web services) and more open markets. However, the establishment of an e-contract is complicated and error prone. There are multiple negotiation styles ranging from auctions to bilateral bargaining. This paper provides an approach for modeling multi-party negotiation protocols in colored Petri nets. It is shown how different negotiation styles can be modeled in a unified and consistent way. Moreover, CPN Tools is used to analyze the resulting colored Petri nets. Simulation can be used for both validation and performance analysis, while state-space analysis can be used to discover anomalies in various multi-part negotiation protocols.

1 Introduction

Contracts are documents that states the mutual obligations and authorizations that reflect the agreements between two trading partners [1]. An *e-contract* is a contract modeled, specified and enacted by a software system [2].

E-contracting has become important because of technological advances, for instance, agent technology and web services have allowed for the establishment of virtual enterprises or management of supply chains. E-contracting is a means to regulate the relationship among partners of a virtual enterprise or a particular supply chain.

According to Angelov [3], automatic negotiation is not a trivial task. Most of the negotiation systems proposed so far have an overly limited scope: they run a particular kind of negotiation – typically a bargain or an auction – in a restricted market place. The produced contracts are signed by only two negotiators. Several bilateral contracts must be produced when the agreement comprises more than two partners. In a collaborative multi-partner settlement, this may cause semantic loss. A single contract signed by several partners can express a common goal shared by them by means of a rich set of complex relationships. Splitting this contract into a number of bilateral contracts can make unclear some of these relationships, and even destroy others. This paper aims at both automatic negotiation and multi-party contracts. It shows how the main parts of the SPICA negotiation protocol [4] can be modeled and simulated by means of Petri nets.

The main contributions are: a) the paper describes a negotiation protocol that combines different negotiation styles — namely: bargain, auction, ballot — to produce an e-contract that can be signed by several partners; b) it shows how such a protocol can be modeled as a colored Petri net (CPN) and simulated using CPN Tools [5]; c) it also shows how the capabilities of CPN Tools assisted in the identification of design flaws and how these were addressed.

The paper is organized as follows. Section 2 overviews some related work. Section 3 briefly introduces various negotiation protocols, presents a running example that is used throughout the paper, and illustrates the SPICA negotiation protocol through short examples. Section 4 shows how the main parts of the SPICA protocol were modeled in terms of colored Petri nets and presents some of its subnets (the total model is composed of 31 subnets). Section 5 describes how the model was adjusted to simulate the running example. Section 6 discusses what we have learnt by modeling and simulating the protocol using CPN Tools, which problems we have encountered, and how they were addressed. Finally, Section 7 concludes the paper.

2 Related Work

This section reviews related work on negotiation processes and the use of CPN Tools to model and validate protocols.

There are a number of mechanisms that guide the negotiation process. Bartolini [6] constructs negotiation templates that specify different negotiation parameters. Chiu [7] also uses contract templates as a reference document for negotiation. Similarly, [8] uses a contract template that describes the negotiation parameters, how they are interrelated, along with meta-level rules about the negotiation. In contrast, [9] uses a set of examples of good agreements and it is up to the negotiator to try to get as close as possible to one of the examples.

Like most of the related work, our negotiation process is guided by a contract model. Thus, broadly speaking, the negotiation process concerns determining values for unbound properties.

Our negotiation process is developed through the exchange of messages among the negotiators that comply with a specific protocol. This is a common approach, like the ones based on FIPA's standards [10], e.g., [11]. Governatori and others [12] use a different approach. They propose a negotiation process that uses Defeasible Logic.

Whatever the basis of the negotiation process, all negotiators must understand concepts and names used during the negotiation. The work of [11] combines the use of ontologies and agent technologies to help in solving naming problems in car assembly supply chain negotiations. Our approach also uses ontologies (see [4])

According to Angelov [3], automatic negotiation is a difficult task. Tools that help in the design and implementation of negotiation protocols are needed. Jensen and others [5] present CPN Tools. It builds on Coloured Petri Nets [13] and provides a rich set of resources for modeling, simulating and validating systems where concurrency, communication, and synchronisation play an important role.

CPN Tools has been used to analyse both experimental protocols and also well-established ones. Chen and others [14] use CPNs to model a multi-agent system for supply chain management based on agent negotiation. They describe their negotiation protocol and explain how they formally modeled the negotiation process by using CPNs. The negotiation protocol presented is similar to our auction and bargain negotiation styles. Liu and Billington [15] analysed the Capability Exchange Signalling (CES) protocol using CPN Tools and found it may fail in some specific circumstances.

3 Negotiation: A Particular Protocol

3.1 Overview of Negotiation Protocols

Negotiation is a process by which autonomous entities communicate and compromise to reach an agreement on matters of mutual interest while maximizing their individuals utilities [16]. Automated negotiation (*e-negotiation*) is done by software rather than people and organizations, in general in the form of software agents, and their agreement is stated in *e-contracts*.

A negotiation process involves a number of issues. To begin with, the *number of negotiators* involved is typically one-to-one (*bargaining*), one-to-many (*bidding*), or many-to-many (*double-auction*). In bargains, one seller deals with one buyer. Bidding also has a broad range of flavors, which involve *request for proposals* and interactions among seller(s) and buyer(s). The so-called English or Dutch auctions have many buyers competing for a product sold by an auctioneer. In English auctions, the auctioneer defines a start price and the bidders increase continuously the item's value until the last bid cannot be beaten. Dutch auctions start with a high value and the auctioneer decreases this value until a buyer agrees to pay the proposed value. A many-to-many negotiation can occur through a so-called *double auction*. In this case, several sellers offer their products on a "shared space" and several buyers submit their bids. There is a mechanism that matches offers and bids.

A second issue that needs to be addressed by the negotiation process is related to the number of items. The negotiation process may be restricted to a single product or to a bundle of items. In the latter case, the buyer may be interested in buying all items or none of them [8].

A third issue refers to the negotiation strategy. According to [12], techniques for designing negotiation strategies can be classified into three categories: (i) game-theoretic, (ii) heuristic, and (iii) argumentation. The first approach models a negotiation situation as a game and attempts to find dominant strategies for each participant by applying game theory techniques. In heuristic-based approaches, a strategy consists of a family of tactics (i.e., a method for generating counter-offers), and a set of rules for selecting a particular tactic depending on the stage of the negotiation. Argumentation-based approaches extend heuristic ones by introducing communication patterns such as threats (e.g., "that is my last offer"), rewards, etc.

The fourth issue concerns how the negotiation process is finished: either by agreement or by withdrawal. In [6], agreement formation rules determine which proposals are matched. In [9], a negotiation is aborted if agreement is not reached in a predefined number of rounds.

Finally, the last issue concerns how to renegotiate an existing contract due to some external change. This includes discovering which contracts were affected by the change, which clauses should be modified, who should modify the contract, and so forth.

3.2 Running Example

This section presents the running example that will be used throughout the remainder of the paper.

Milkyway Dairy is a fictitious dairy industry that produces milk derivatives, especially lowfat yoghurt. The farms in its region mainly produce milk and fruits. Periodically, Milkyway Dairy negotiates contracts with nearby farmers for future delivery of milk, blueberries and peaches. Thus, they can plan their future production. Because of governmental regulations, the dairy and the farmers must agree upon maximum milk quota each one will be allowed to deliver. However, the dairy will seek the cheapest supplier for each fruit. The price of the milk is defined at delivery by an official price list, but Milkway negotiates with the farmers a minimum price that will be payed at delivery. In turn, the farmers guarantee that they will deliver at least 500 liters of milk each week. The prices and quantities of different fruits are established during the negotiation.

Milkyway has a predefined draft (or model) for this contract. The negotiation process consists of filling some gaps in this model. The gaps, denominated *properties*, are identified by names preceded by # (e.g., #MINPRICE, #PRICEBERRY). A simplified version of this contract model is shown in plain English in Figure 1. Thus, the parties will agree upon, for instance, a value for the minimum price (#MINPRICE in Clause 3) and which farm will provide berries (#Fa). Note that this contract is a *multi-party contract* signed by the dairy and by several farmers.

> **Clause 1:** Milkyway Dairy, herein named Dairy, and farms Farm 1, Farm 2, Farm 3, Farm 4, herein respectively named F1, F2, F3 and F4, agree that the amount of milk Dairy may buy each week from any of the stated farms is at most *#QMAX* liters.
>
> **Clause 2:**F1,F2,F3 and F4 will deliver altogether at least 500 liters of milk per week to Dairy.
>
> **Clause 3:**The price of milk is defined by the Official Price List at the moment of delivery. However, the Dairy agrees to pay at least *#MINPRICE* per liter of delivered milk.
>
> **Clause 4:**If farms F1, F2, F3 and F4 do not succeed to fulfill clause 2, they must pay Dairy a fine worth 10000**#MINPRICE*.
>
> **Clause 5:**Farm *#Fa* agrees to deliver *#QBERRY* Kg of blueberries at €*#PRICEBERRY* a kilogram weekly.
>
> **Clause 6:** Farm *#Fb* agrees to deliver *#QPEACH* Kg of peaches at €*#PRICEPEACH* a kilogram weekly.

Fig. 1. *Simplified multi-party contract with six properties that need to be negotiated*

3.3 The SPICA Protocol

This section describes the negotiation process. In this paper, we focus on our SPICA protocol [4]. SPICA (SuPply chain Integration, Coordination, contracting and Auditing framework) is the Latin term for the ear of some grains (e.g., as in ear of corn, or wheat). It is the main star in the Virgo constellation. This constellation is associated to a few myths related to agriculture. In this protocol, the negotiation process is orchestrated by a leader negotiator and is guided by a contract model. The contract model is a predefined contract template which is filled in with values agreed upon by the negotiators. After a successful negotiation, a new contract is created from the model and the negotiated values. Subsequent sections detail the negotiation process itself.

Organization of the Negotiation. A negotiation process involves two or more negotiators. One of them is the *leader*. There is a *notary* responsible for bureaucratic chores (e.g., constructing the final contract) or acting as a trusted third-party (e.g., to control ballots).

These players exchange information within a negotiation process through asynchronous messages. The messages may be peer-to-peer or broadcasted. A contract negotiation has several phases, including leader election, objective and restriction announcement, property negotiation, commit attempt, and contract construction. These phases are described in [4], the core being property negotiation. This is the focus of this paper and is presented in the following sections.

Core Negotiation. There are three generic styles of negotiation: *ballots*, *auctions* and *bargains*. Ballots are used when the negotiators have to reach consensus on a property's value. In our example, most of the farms must agree upon a value for QMAX. Auctions are used when there is competition among different negotiators in order to bind a property to a value. This is the case of property PRICEBERRY. Bargains are used when there are two negotiators and they want to interact to reach a value that is convenient for both. For instance, in our example the dairy and Farm 4 interact to find a consensual value for PRICEPEACH. All these styles may be used to negotiate a single contract.

The negotiation styles are built on a few negotiation primitives, i.e., types of messages exchanged among the participants. These primitives rely on two basic mechanisms: *request for proposals* (RFP) and *offers*. The following paragraphs describe RFPs and offers. Subsequently, the negotiation primitives and the negotiation styles are shown in a few short examples.

An *RFP is an invitation.* A negotiator A sends an RFP to a negotiator B asking for a value for one or more properties. An RFP may prescribe some restrictions on the expected answer and may also bind the value of other properties. For instance, the dairy intends to buy 100kg of peaches from Farm 4. Thus, it sends an RFP to Farm 4 assigning the value 100 to QPEACH and asking a value for PRICEPEACH.

An *offer is a promise.* A negotiator who wants to assign a value to one or more properties sends an offer to another negotiator. The offer indicates the properties the first negotiator is interested in and the values it proposes for them. If the other negotiator accepts such offer, both negotiators are committed to the proposed values. A negotiator can answer to an RFP by sending back an offer that proposes values for the desired properties and that complies with the restrictions indicated in the RFP. If a negotiator is not interested in a RFP, it replies to the RFP by indicating that it will not send an offer. Conversely, a negotiator who receives an offer agrees upon it, disagrees, or proposes a counter-offer. Thus, two negotiators may engage in a cycle of counter-offers until they reach an agreement or give up.

To sum up, offers and RFPs have two distinguishing differences. First, all properties included in an offer must have their values defined in advance, the other party may only agree or disagree on those values. In contrast, an RFP may have properties with predefined values, but also, at least, one unbound property, whose value is subject to negotiation. Second, the party that issues an offer is committed to it, i.e., if the other party agrees upon, the issuer must honor it. In contrast, the party that issues an RFP is neither obliged to keep the RFP valid nor to accept any offer in response to it. That is, an RFP does not imply any level of commitment.

The following describes how RFPs and offers are exchanged by means of specific messages, and how such messages are organized into the three negotiation styles mentioned before (bargains,

ballots and auctions). In these examples, any text enclosed by triangles represents an RFP, and text enclosed by squares represents an offer. They do not show all the information they carry, but only the information relevant for the examples. Messages are numbered and show the sender (e.g., `dr:`) for convenience.

Figure 2 shows the dairy (`dr`) and Farm 4 (`f4`) bargaining the price of a certain amount of peaches. In the first message (1), the dairy sends an RFP to farm `f4` asking for a proposal for property PRICEPEACH, considering that property QPEACH has value 100. Next, (2) `f4` offers 108 for that property. However, the dairy does not agree upon such value and (3) sends a counter-offer proposing 92. Finally, (4) `f4` agrees upon the proposed value. This sequence of counter-offers may be arbitrarily long and may also be finished without agreement by sending a `proposal no_agree` message.

```
1. dr: send f4  new_rfp ◁PRICEPEACH?; QPEACH=100▷
2. f4: send dr  new_offer □PRICEPEACH=108; QPEACH=100□
3. dr: send f4  new_offer □PRICEPEACH=92; QPEACH=100□
4. f4: send dr  proposal agree □PRICEPEACH=92; QPEACH=100□
```

Fig. 2. *Bargain*

The next two examples show how the notary cooperates in the negotiation process. Figure 3 presents the negotiation of property `QMAX`. This property requires that most of the farms agree on a value, thus a ballot is performed under control of the notary. First, the dairy asks the notary to conduct the ballot. It informs the issue to be voted and the possible votes. In the example, the issue is an offer proposing the value 187 to property `QMAX` and the alternatives are agree (`VOK`) or disagree (`VNOK`). Next, (2) the notary (`nt`) acknowledges to the leader (i.e., `dr`) that it will conduct that ballot. The notary (3) broadcasts the issue to be voted to the negotiators. Each negotiator (4-7) sends its vote to the notary. Finally, the notary counts the votes and (8) broadcasts the result informing that the issue was approved and the number of votes each alternative has received. There are other messages to inform that the issue was not approved or that a tie has happened.

```
1. dr: send nt  control_ballot □QMAX=187□ VOK,VNOK
2. nt: send dr  will_conduct □QMAX=187□ VOK,VNOK
3. nt: broadcastvoting □QMAX=187□ VOK,VNOK
4. f1: send nt  vote VOK
5. f2: send nt  vote VOK
6. f3: send nt  vote VNOK
7. f4: send nt  vote VOK
8. nt: broadcastbal_res BAPPROVED VOK:3, VNOK:1
```

Fig. 3. *Ballot*

The scenario shown in Figure 4 follows a variant of an English auction that aims at minimizing the value of `PRICEBERRY`. The maximum price is 180. Thus, (1) the dairy asks the notary to advertise an auction for an RFP and wants the notary to collect at most four answers within 30 seconds. The notary (2) broadcasts the RFP and waits as requested. The farms receive the RFP and (3-6) send offers to the notary in response. A negotiator may decline an RFP by returning a `no_offer` answer. The notary collects the answers and (7) sends them to the dairy. The labels O_1,...,O_4 stand for the offers from `f1`,...,`f4`, respectively. Now, the leader chooses the best offer (`PRICEBERRY`=112) and (8) asks the notary to start a new auction round, restricting the expected price. This process is repeated (9-12). Eventually none of the negotiators is interested in offering a lower price and all of them answer `no_offer`. Now, the dairy (13) agrees upon the best offer of

the previous round. The dairy also sends a `proposal no_agree` message to all defeated offers (not shown in the figure).

```
1. dr: send nt   first_answers 4 30 ◁PRICEBERRY?≤180; QBERRY=100▷
2. nt: broadcastnew_rfp ◁PRICEBERRY?<180; QBERRY=100▷
3. f1: send nt   new_offer □PRICEBERRY=160; QBERRY=100□
4. f2: send nt   new_offer □PRICEBERRY=112; QBERRY=100□
5. f3: send nt   new_offer □PRICEBERRY=130; QBERRY=100□
6. f4: send nt   new_offer □PRICEBERRY=170; QBERRY=100□
7. nt: send dr   collected_answers O1,O2,O3,O4
8. dr: send nt   first_answers 4 30 ◁PRICEBERRY?<112; QBERRY=100▷
9. nt: broadcastnew_rfp ◁PRICEBERRY?<112; QBERRY=100▷
10. f2: send nt   no_offer
11. f3: send nt   new_offer □PRICEBERRY=98; QBERRY=100□
12.              ...
13. dr: send f1   proposal agree □PRICEBERRY=47; QBERRY=100□
```

Fig. 4. *Auction*

The examples given in this section show how the properties of our running example (Figure 1) can be negotiated using different styles of negotiation. However, the style of negotiation has only been described in plain English and can be interpreted in different ways. Therefore, we formalize the different styles and primitives in terms of CPNs.

4 Modeling the SPICA Negotiation Protocol in CPNs

This section describes how the protocols presented in Section 3.3 can be modeled as Colored Petri Nets (CPNs) using CPN Tools. The Petri net model is composed of 31 subnets (19 if replicated instances are not counted). The top-level page is presented in Figure 5. In total there are about 300 places and 200 transitions. This model assumes the absence of exceptions (e.g., communication infrastructure is reliable) and the diligence and fairness of the negotiators. Thus, the negotiators will always receive and answer properly any message from the leader or the notary. The model also makes some simplifications. For instance, RPFs and Offers can be used to announce an auction, however, the model only accepts RFPs.

Section 4.1 presents the most important color sets used in the model. Section 4.2 shows a model for the overall negotiation process. Section 4.3 details the subnets for bargains. Finally, the models for ballots and auctions are only briefly described because of space limitations (Section 4.4).

4.1 Color Sets

This section presents the main color sets used in the model. The names of the properties in the contract model are represented by the `PropertyName` color set. The color set `NegId` is used to identify and to address negotiators: `L` stands for the leader and `F1`...`F4` refer to the farms; `EVERYBODY` stands for all farms; `NO_BODY` is an invalid value for initialization purposes.

colset PropertyName = **with** QMAX | MINPRICE | QBERRY | PRICEBERRY |QPEACH |PRICEPEACH;
colset NegId = with L | F1 | F2 | F3 | F4 | EVERYBODY | NO_BODY;

The `Property` color set represents the properties being negotiated. It consists of a property name (`PropertyName`), a status code that indicates if it was agreed upon (`PropAgreement`), the `Value` agreed upon (if any), and the list of the negotiators who had agreed upon this value `LstNegId`. `PropAgreement` is set to `AGREED` to indicate that the property was negotiated and the partners have agreed upon a value; `NOT_AGREED` is the opposite, and `IN_NEGOTIATION` indicates that the property was not negotiated.

```
colset LstNegId = listNegId;
colset PropAgreement = with AGREED | NOT_AGREED | IN_NEGOTIATION;
colset Value = INT;
colset Property = product PropertyName * PropAgreement * Value * LstNegId;
```

The leader builds a negotiation plan (`NegoPlan`) before it starts negotiating the contract's properties (Section 4.2). A negotiation plan is a table whose entries describe which properties will be negotiated together (`PropertyNameLst`) and the style of negotiation to be used (`TypeNego`). For instance, in our example, properties `QBERRY` and `PRICEBERRY` are negotiated together using an auction (`AUC`). Other styles of negotiation are ballots (`BLT`) and bargains (`BARG`).

```
colset TypeNego = with BLT | AUC | BARG;
colset PropertyNameLst = list  PropertyName;
colset NegoPlanItem = product TypeNego * PropertyNameLst;
colset NegoPlan = list  NegoPlanItem;
```

Negotiators exchange RFPs and offers within a negotiation. An RFP has a unique identifier (`RfpId`), and references to the originator (`From`) and recipient (`To`) of the request. The recipient may be a single negotiator (e.g., `F1`) or `EVERYBODY` which causes the RFP to be broadcasted.

```
colset RfpId = MessageId;
colset From = NegId;
colset To= NegId;
colset Operator = with GE | LE | EQ | OM | NEQ;
colset Restriction = product PropertyName * Operator * Value;
colset LstRestriction = list Restriction;
colset Rfp = product RfpId * From * To * LstRestriction * AuctId;
```

The RFP also has a list of restrictions (`LstRestriction`) on the properties values. For instance, the list of restrictions for the RFP of the initial message in Figure 4 should be:

```
[ (QBERRY,EQ,100), (PRICEBERRY,LE,180), (PRICEBERRY,OM,0) ]
```

Note that OM stands for "offer me", and the value after it is meaningless.[4] Thus the negotiator is expecting an offer for `PRICEBERRY`, provided that it is less or equal (`LE`) than `180`. The other operators are great or equal (`GE`), equal (`EQ`), and not equal (`NEQ`). The value for `QBERRY` is already bound by the RFP and the recipient must agree on it. An RFP may be used in an auction. If so, the auction identifier (`AuctId`) is also set.

An offer may be an answer to a previous RFP or to another previous offer. Thus, `ParentId` relates the offer to the previous RFP. The answer to an RFP can be a *No Offer*. This means that the negotiator is not interested in sending an offer for such RFP. In this case, `NoOffer` is set. The offer identifies its originator and recipient to be an RFP (`From` and `To`). When a negotiator receives an offer, it evaluates it. If the negotiator agrees upon the received offer, it assigns the value `OK` to field `Eval`; otherwise, it assigns the value `X` to it. In an auction, the best offer of the current round is marked with `NR` (New Round), indicating that a new auction round should start. When an offer is created, the `Eval` field is set to `NE` (Not Evaluated). `PropertyLst` has all the properties being negotiated. This list must contain all properties referenced by a previous RFP (if any). The negotiator that issued the offer may accept a subsequent counter-offer by setting `CounterOfferAllowed` to `true`. Note that in an auction (Section 4.4) it is always set to `false`. `ParentIsRfp` is set to `true` whenever the offer answers an immediate previous RFP. Finally, an offer has a unique identifier (`OfferId`).

```
colset ParentId = MessageId;
colset Eval= with OK | X | NE | NR;
colset NoOffer = int with  1..0;
colset PropertyLst = listProperty;
colset CounterOfferAllowed = BOOL;
colset Offer = product ParentId * NoOffer * From * To * Eval * PropertyLst * CounterOfferAllowed
* ParentIsRfp * OfferId;
```

[4] Note that the union type could be used here. However, for reasons of simplicity we did not do so.

Other colors used in the model that are specific to different negotiation styles will be described when needed.

4.2 The Overall Negotiation Process

Figure 5 shows the main net of our model. The negotiation process aims at assigning values to properties. The properties to be negotiated are in the place `ToNegotiate`. Broadly speaking, the leader builds a negotiation plan (see transition `PlanHowToNegotiate`). Then, the leader follows this plan and coordinates the negotiation. At the end, properties are grouped based on whether they were agreed on or not (see places `Agreed` and `NotAgreed`). Some properties may fail to be negotiated, e.g., when no negotiator is interested in an auction. Such properties are put in the place `NotNegotiated`. If all properties were successfully negotiated, a new contract can be built. This last step is out of the scope of this paper.

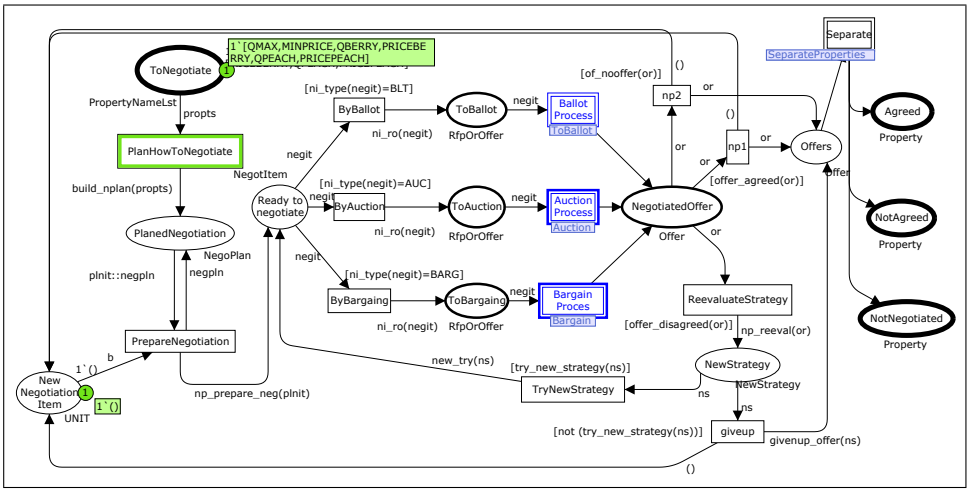

Fig. 5. *Negotiation Process (NegotiateProperty page)*

The negotiation plan for our example is shown in Figure 6. It shows that the leader will first negotiate property `QMAX` using a ballot, then `MINPRICE`. In the third step, it will negotiate properties `QBERRY` and `PRICEBERRY` by an auction. Finally, `QPEACH` and `PRICEPEACH` will be negotiated through a bargain. Note that each step uses one negotiation item and starts after the previous one has ended (notice place `NewNegotiationItem`).

Each negotiation style (bargain, ballot, or auction) is started by sending the first RFP or offer to the corresponding subnets (substitution transitions `BallotProcess`, `AuctionProcess` and `BargainProcess`). To enable reuse, a new color set (`RfpOrOffer`) is introduced and used in places where tokens represent RFPs or offers. The negotiation is carried out by the three subnets shown in Figure 5. These subnets deliver back the offers that were agreed upon and those that were not. In the latter case, the leader may change the negotiation strategy (transition `ReevaluateStrategy`), even changing the negotiation style, and submit them again to be negotiated.

	Style	Properties
1.	BLT	[QMAX]
2.	BLT	[MINPRICE]
3.	AUC	[QBERRY,PRICEBERRY]
4.	BARG	[QPEACH,PRICEPEACH]

Fig. 6. Negotiation plan for the running example

Figure 7, shows the tree view of all subpages. Figure 7.a depicts the topmost net and the three main subnets. Figures 7.b to 7.d detail these subnets. The nodes are named after the page names, and a number in front of the name denotes the number of replications of this page (if any). For instance, there are four replications of `ProcessRfp` within page `AuctionNegotiators` (Figure 7.b). Each node is annotated with a pair `P:X T:Y`. The first stands for the number (`X`) of places (`P`) in that page and the latter, the number (`Y`) of transitions (`T`), e.g., the page `BargainNegotiators` has 10 places and 6 transitions ((Figure 7.d)). The paper only shows the underlined pages.

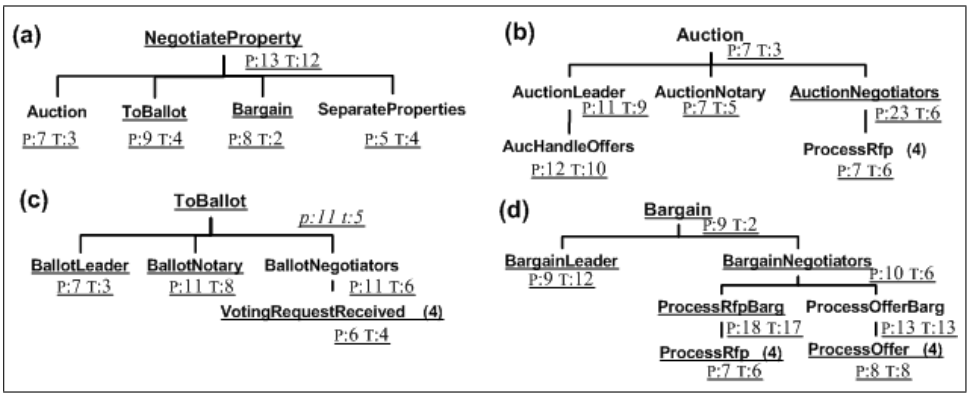

Fig. 7. *Tree view: (a) topmost page and its main subpages; (b)-(d) subpages are detailed*

4.3 Modeling Bargains

A bargain is characterized by a cycle of offers and counter-offers. The negotiators end up agreeing upon an offer or they give up. Figure 8 shows the subnet that models the bargaining process. The RFP or offer to be bargained enters the subnet via place `ToBargain`. The leader directs this RFP (or offer) to the other negotiators by putting it in place `RfpToNegotiators` (or `OfferToNegotiators`). The negotiators can make counter-offers by putting offers in the place `OfferToLeader`. Leader and negotiators may engage in a cycle of counter-offers until one of them agrees upon the offer or gives up the bargain. In both cases, the partner places an *acceptance notification* in the place `NotificationToNegotiators` or `NotificationToLeader`. An acceptance notification is an offer with special settings, which is denoted by the `Eval` field. At the end, the leader puts the negotiated offers in the place `OfferNeg`.

Figures 9 and 10 detail the substitution transitions `Leader` and `Negotiators`, respectively. Figure 9 shows how the leader develops the bargain and Figure 10 shows how the negotiators react to it. The latter figure also shows how RFPs and offers are processed in a bargain. Figure 9 shows a few design directives used in the model. First of all, the messages were modeled as transitions. The firing of such transitions means that a message is sent, broadcasted, or received. The transitions are named after the message. Prefixes are used to identify whether the data was sent (`s_`), broadcasted (`b_`) or received (`r_`). For instance, the transition named `b_new_rfp` (Figure 9) means that the message `new_rfp` was broadcasted to the negotiators. There are transitions not related to exchange of messages. The second design decision that should be noted is that data conveyed by messages are modeled as tokens, e.g., the place `ToBarg` (Figure 9) stores the RFP or offer that starts the bargaining process.

When there is an RFP to be negotiated in place `ToBarg` (Figure 9), the leader sends or broadcasts a `new_rfp` message that conveys such RFP. This is represented by transitions `s_new_rfp` and `b_new_rfp`, respectively. After a few firings, such an RFP will be available to the other negotiators in the place `RFP` (Figure 10). Similarly, when there is an offer to be negotiated, the leader sends or broadcasts it. It will be available to the negotiators via place `OFFER_fr_LEADER` (Figure 10).

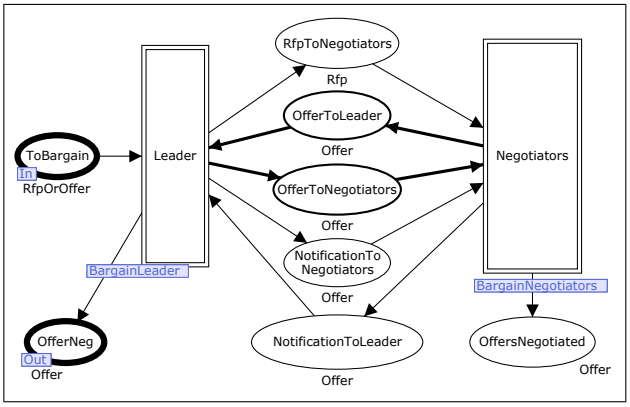

Fig. 8. *Bargain subnet. Cycle of counter-offers is emphasized.*

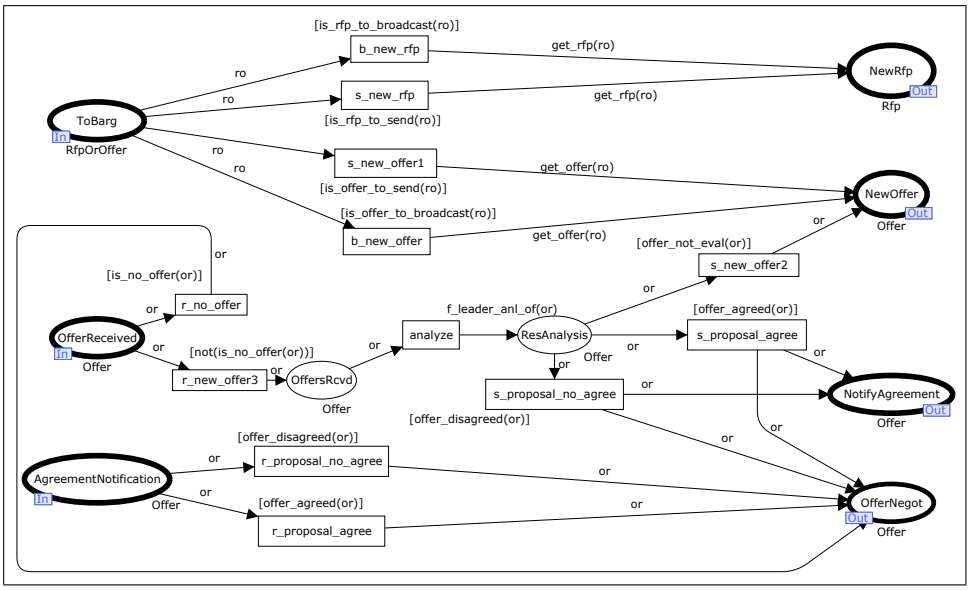

Fig. 9. *BargainLeader subnet*

When the other negotiator(s) receive(s) an RFP (in the place `RFP`, Figure 10), the answer to the leader is put in the out-port `OFFER_to_LEADER`. The negotiator(s) receive(s) offers from the leader through the in-port `OFFER_fr_LEADER`. Such an offer may be a response to a directly preceding RFP. In this case, the offer passes through the `Process RFP (Bargain)` subnet, but is handled by the `Process Offer (Bargain)` subnet. The arriving offer should also be a new offer or a counter-offer from the leader. In both cases, the offer goes directly to the `Process Offer (Bargain)` subnet.

An agreement notification from the leader arrives via the port `AgrNotfFrLeader` and may follow two paths: (a) it is directed to `Process RFP (Bargain)` subnet, when the leader immediately agreed upon the first offer sent by the negotiator, or (b) the notification is directed to the `Process Offer (Bargain)` subnet.

Figure 10 shows neither individual negotiators nor makes a distinction whether the messages were sent or broadcasted. These details are presented in Figure 11. An RFP is directed to the corresponding receiver or broadcasted to all negotiators. Note that the answers of each negotiator are joined in one respective single place. The subnet for `Process Offer (Bargain)` is designed in a similar way.

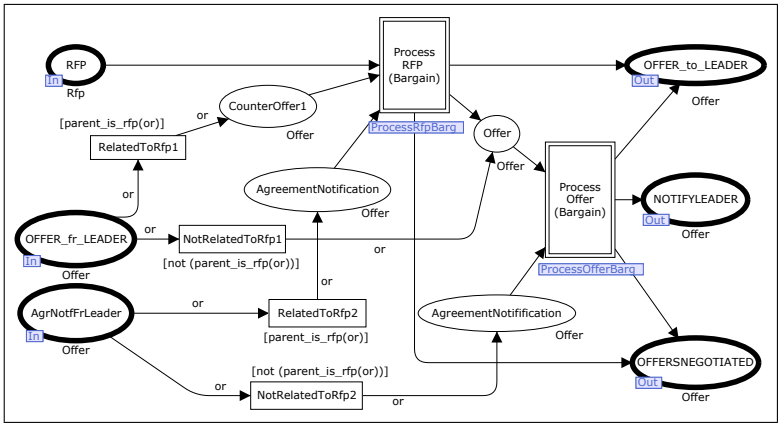

Fig. 10. *BargainNegotiators subnet*

Figure 12 presents the elementary `ProcessRFP`. It shows the messages that are received and sent by the negotiator and the function it uses to assess the received RFP (`f_analyze_rfp`). Recall that this subnet is used by several negotiators and each negotiator evaluates the RFP differently. Thus, the `f_analyze_rfp` function actually dispatches the RFP to specific functions. For this reason, the token in `RFP` place is of color `RfpXNegId` and contains the RFP and the identification of the negotiator it is meant for.

Finally, Figure 13 shows the elementary `ProcessOffer`. New offers arrive in the place `Offer`. Function `f_analyze_offer` dispatches it to the correct negotiator. The (dis)agreed offers are made available to the partner negotiator through the place `AgreeDisagree`. They are collected in the `OffersNegotiated` place. Conversely, the negotiator decides to make a counter-offer, which is placed in `CounterOffer`. A `No Offer` message is just collected (transition `r_no_offer`). Agreement notifications related to previous offers are received in through port `AgreementNotification` and are also collected.

Note that the same patterns of `ProcessRfp` and `ProcessOffer` were applied to subpages at different levels. For instance, Figures 10 and 11 have the same structure of Figure 12.

4.4 Modeling Ballots and Auctions

Figure 14 details the `BallotProcess` subnet referred to in Figure 5. It shows the leader, the notary and the negotiators interacting in a ballot process. The leader role is detailed in Figure 15. Note that the leader asks the notary to conduct the ballot (transition `s_control_ballot`), receives the notary's acknowledgment (transition `r_will_conduct`) and, eventually, receives the result of the ballot (`r_bal_res`).

The notary role is detailed in Figure 16. First, the notary receives a message from the leader to conduct a ballot (transition `r_control_ballot`). It accepts the job (transition `s_will_conduct`) and broadcasts the issue to the negotiators (`b_voting`). The notary receives votes (`r_vote`) or vetoes (`r_veto`). In case it does not receive any veto, the notary counts the votes and broadcasts the result (`b_bal_res`). Otherwise, the notary informs all participants that a veto has occurred. However, none of the farms in the running example has veto power. Thus, the veto scenario does not occur.

The BallotNegotiators subnet (not shown) directs the issue to be voted to each negotiator, collects their votes and, eventually, sends them the ballot result. Figure 17 shows how a negotiator reacts when it receives a vote request. Basically, it analyzes the issue and decides either to vote or to veto (see function `vrr_analyze`). Its choice is sent to the notary (`s_vote` or `s_veto`). Eventually, the negotiator receives the ballot result (`r_bal_res`).

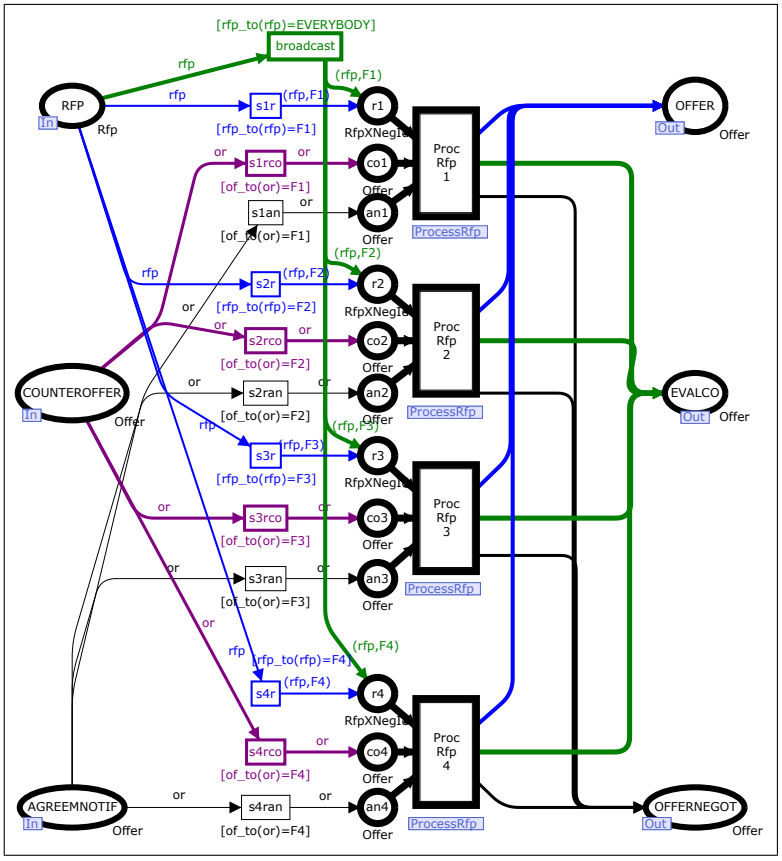

Fig. 11. *ProcessRfpBarg subnet*

The notary also collaborates in an auction. Figure 18 shows an auction step being directed to negotiators. Each negotiator uses ProcessRfp subnet to react to a negotiation step. Note that this is the same subnet used in bargains. The other subnets are not presented for the sake of brevity.

5 Implementing the Negotiator's Strategies

To simulate the model, we configured the CPN for the scenario presented in Section 3.2. For each property, each negotiator (including the dairy) has a value (or a range of values) it considers a "good deal" and for which it will always agree upon. There are also values the negotiator will never agree upon. Values in-between will be accepted with specific probabilities: the nearer to the expected value, the higher the acceptance probability. Each farm has also distinctive characteristics, mainly its production capability for each product (Figure 19). These characteristics influence its decision. For instance, Farm 1 weekly produces a large amount of milk (400 liters). Thus, it tends to agree upon high values for `QMAX`. However, Farm 2 has a low production capacity (50 liters). It will not agree upon high values for `QMAX` for fear that the week quota should be fulfilled by the other farms.

According to the model presented in Figure 5, the leader may have several strategies to negotiate the value of a property. When a negotiation fails, the leader can try again negotiating that property using a different strategy. The simulation uses this capability to negotiate properties `QMAX` and `MINPRICE`. For instance, the leader has a lower and an upper limit to `QMAX` (`QMAX_min` and `QMAX_max`). It starts in the middle point between these values and submits increasing values to successive ballots. If the proposed values reach unsuccessfully the upper limit, the leader submits successive decreasing values starting from the middle point.

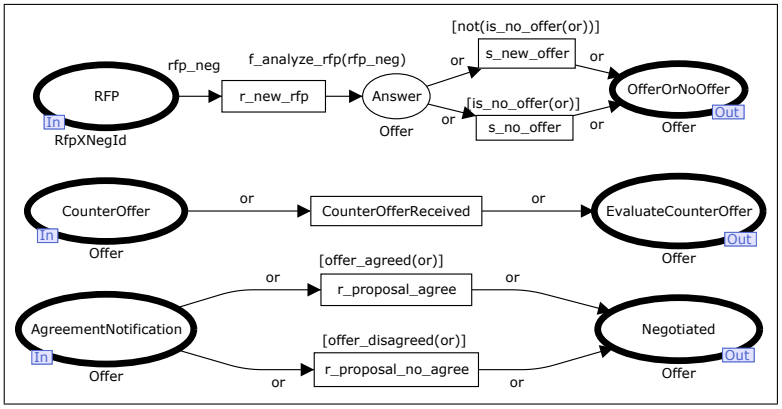

Fig. 12. *Elementary Process RFP (ProcessRfp subnet)*

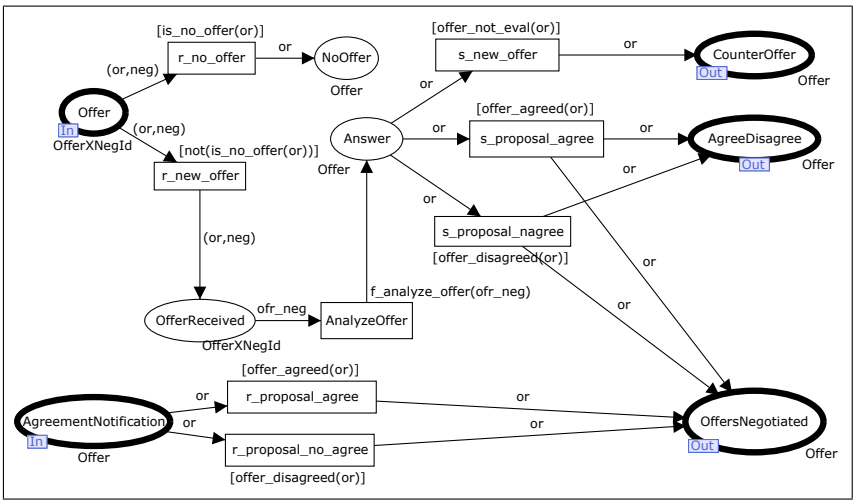

Fig. 13. *Elementary Process Offer (ProcessOffer subnet)*

The dairy is interested in buying 100kg of peaches weekly. Only Farm 4 can afford such amount. Thus, they engage in a bargain that comprises two properties: `QPEACH` (bound by the leader) and `PRICEPEACH`. The dairy has two parameters that guide the negotiation. `PEACH_PRICE_GD_LD` is a threshold value, i.e., values lesser or equal to it are considered a good deal (cheap enough) and are always accepted. `PEACH_PMAX_LD` is the opposite, values higher than it are refused.

The dairy starts the bargain by sending an RFP to Farm 4 assigning the value of 100Kg to QPEACH and requesting a value for `PRICEPEACH`. Eventually, the dairy receives an offer from Farm 4. The closer the offered price is to the "good deal", the higher the probability of being accepted. If not accepted, the dairy will choose (by chance) between refusing the offer or making a counter-offer. If it chooses to make a counter-offer, the value for a future counter-offer uses the value of `PEACH_PRICE_GD_LD` (e.g., 10) as a reference point: if the proposed value (e.g., 14) is at a "distance" `d` (i.e. 4), beyond the "good deal", the counter-offer will be exactly at the same "distance" before the "good deal" (i.e., 6).

Similarly, Farm 4 has two parameters to negotiate `PRICEPEACH`. The value it considers a good deal is `PEACH_PRICE_GD_F4`. Values higher than it are always accepted. `PEACH_PMIN_F4` is the opposite (too cheap), values lower than it are always rejected. The diagram in Figure 20 depicts Farm 4's decision table. `GD` stands for `PEACH_PRICE_GD_F4`; `MP`, for `PEACH_PMIN_F4`; numbers above the line are fractions (in percentage) of `GD`; and numbers below the line are acceptance probabili-

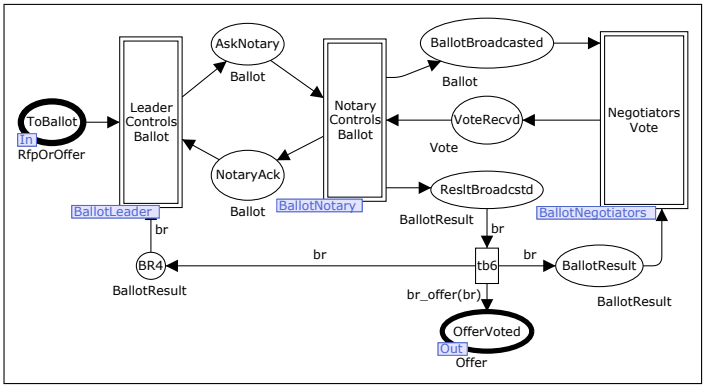

Fig. 14. *BallotProcess subnet*

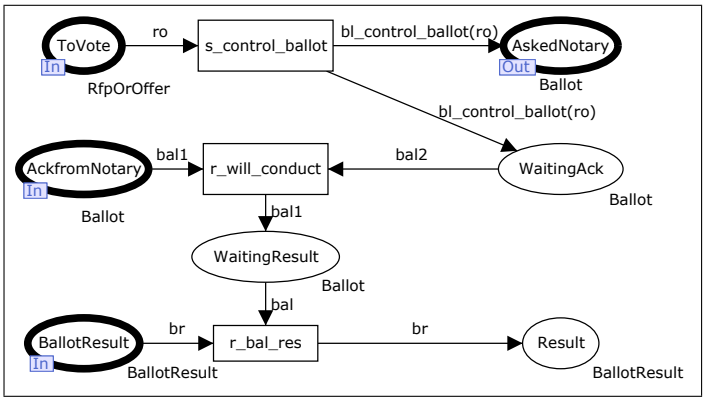

Fig. 15. *BallotLeader subnet*

ties. For instance, if Farm 4 receives an offer whose value is at least 90% of `GD`, but no bigger than `GD`, it will accept the offer with probability `0.8`.

Other negotiators use similar approaches, but with different parameter values. They are not described here for brevity.

Recall Figure 5. One execution of the model should produce as a result of a negotiation the markings shown in Figure 21. Note the agreed values for properties `MINPRICE`, `QMAX` and `PRICEBERRY`. There was no agreement on the value for `PRICEPEACH`.

6 Discussion

This section discusses what we have learnt from the modeling process and some results obtained from simulations and state space analysis.

The construction of the model led us to new insights into the modeling process and into our negotiation protocol. We followed a top-down approach to model the protocol. At the end, we noticed that there were different subnets to handle RFPs in auctions and in bargains. This also happened to offers. Thus, we updated our model using a bottom-up approach: we focused on the handling of RFPs and offers and how to combine them to allow different styles of negotiation (ballots, auctions, and bargains), e.g., compare Figures 11 and 18. This gave us a better understanding about the role of RFPs and offers within our protocol. This allowed us to replicate elementary structures in higher subnets. For instance, compare Figures 10, 11 and 12. Note that they have identical interfaces, and the flow of information is the same.

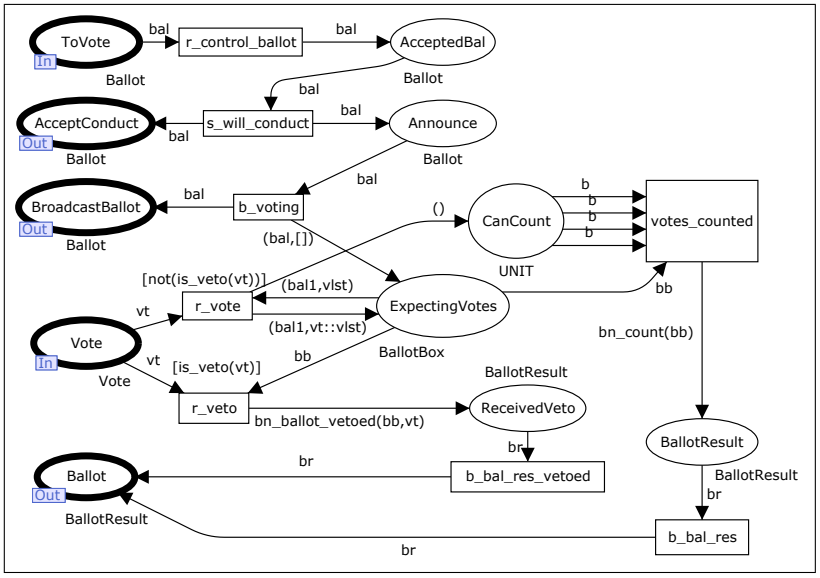

Fig. 16. *BallotNotary subnet*

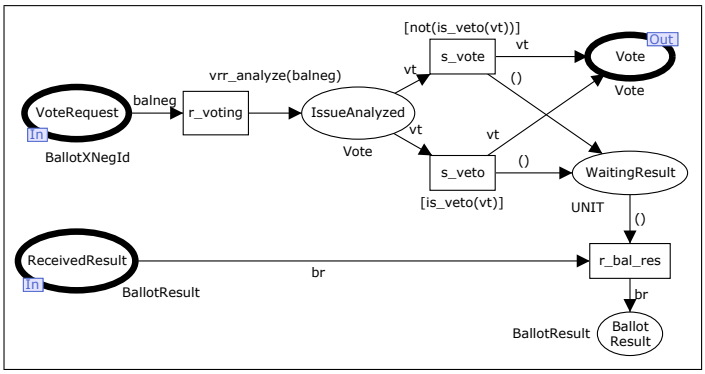

Fig. 17. *VotingRequestReceive subnet*

The modeling process also helped us to improve the SPICA protocol. It showed that the protocol lacked negotiation plans and also that it should allow alternative strategies to negotiate a property.

We used CPN Tools to build the model. CPN Tools helped us in several ways. First, calculating the model's state space was a hard job. Several simplifications and processing hours would be needed to produce a full state space. This made clear that the protocol's implementation will demand extreme care. Second, it helped us to assess the correctness of the model, especially by indicating some unexpected dead markings. Third, it allowed to simulate the running example presented in Section 3.2.

State space analysis. The calculation of the state space was a non-trivial task for several reasons. The model's first version aimed at producing actual values for the properties, e.g., a negotiation should come out with the value 187 for property QMAX in our running example. This easily produces a huge state space. We tried several alternatives to calculate a full state space. First, we created a new model version with some simplifications. The allowed values for properties were restricted (e.g., -1..+1) and the decisions were not based on the offered values, but taken by chance. However, these restrictions did not make possible the calculation of a full state space, even spending several processing hours of a Pentium 4 (3GHz, 1Gb). Then, we split the model

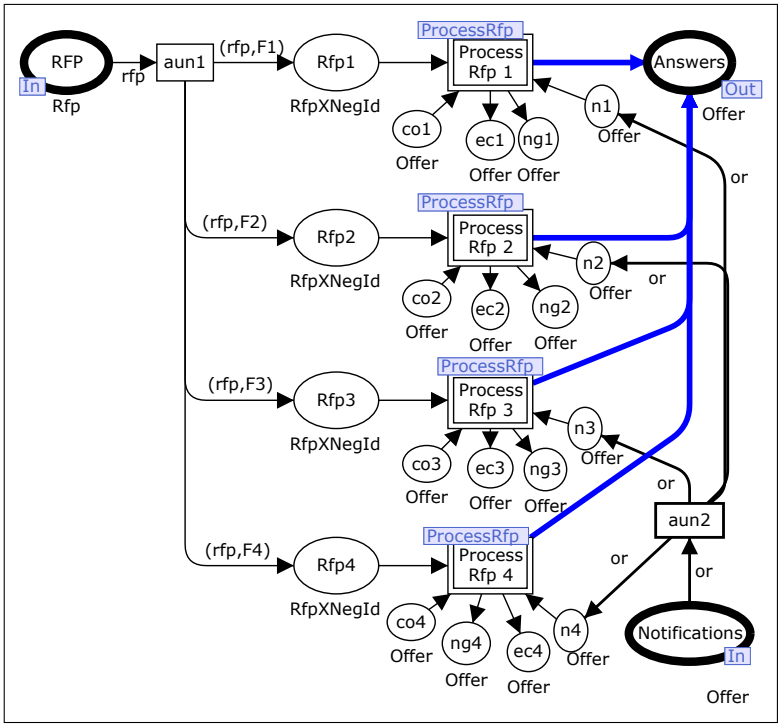

Fig. 18. *AuctionNegotiators subnet*

Production Capacity	**F1**	**F2**	**F3**	**F4**
Milk (liters)	400	50	200	150
Blueberry (kg)	100	200	150	300
Peach (kg)	50	30	80	100

Fig. 19. Production capacity

into tree distinct models – one for auctions, another for ballots and another for bargains – and deleted unneeded places and transitions of the main net (Figure 5).

For the bargain sub-model, we managed to calculate the full state space. However, this was not possible for the ballot sub-model. Thus, we tried to reduce the number of negotiators by dropping two negotiators, i.e., only the other two remaining negotiators took part in ballots. We succeeded in obtaining a full state space for the reduced ballot sub-model (Figure 22). This analysis showed a few dead transitions, which are caused by two factors: (a) the model's reduction and (b) some possibilities allowed by the model were not used by the running example. The dead transitions `SeparateProperties'not_negotiated` and `SeparateProperties'sp3` are examples for the first factor. A property can be *not negotiated* only when a negotiator is not interested in an RFP and sends back a *No Offer* answer. This cannot happen in a ballot. The other transitions are related to the veto power a negotiator may have in a ballot. The farms in our running example do not have such a power.

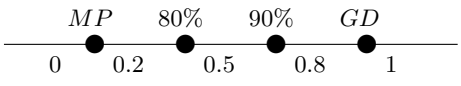

Fig. 20. F4's PRICEPEACH decision table

Agreed	1‘(QMAX,AGREED,202,[ ]) ++ 1‘(MINPRICE,AGREED,15,[ ]) ++ 1‘(QBERRY,AGREED,100,[F3]) ++ 1‘(PRICEBERRY,AGREED,39,[F3])
NotAgreed	1‘(QPEACH,NOT_AGREED,100,[L] ++ 1‘(PRICEPEACH,NOT_AGREED,57,[L])

Fig. 21. A negotiation result

Statistics
State Space
Nodes: 6387 Arcs: 22816 Secs: 26 Status: Full
Scc Graph
Nodes: 6387 Arcs: 22816 Secs: 2

Boundedness Properties
(Omitted)

Home Properties
Home Markings
None

Liveness Properties
Dead Markings
8 [6387,6386,6385,6380,6372,...]
Dead Transition Instances
BallotNotary'b_bal_res_vetoed 1 BallotNotary'r_veto 1
SeparateProperties'not_negotiated 1 SeparateProperties'sp3 1
VotingRequestReceived's_veto 1 VotingRequestReceived's_veto 2
Live Transition Instances
None

Fig. 22. *State space report for the ballot sub-model (excerpt).*

For the auction sub-model, even making such reductions, it was not possible to produce a full state space.

Statistics
State Space
Nodes: 241569 Arcs: 1134868 Secs: **43000** Status: Partial
Scc Graph
Nodes: 241569 Arcs: 1134868 Secs: 190

Fig. 23. *State space report for the auction sub-model (excerpt).*

Model correctness. CPN tools helped us to assess the correctness of the model. It showed a few expected dead markings (e.g., in those states that collect negotiated offers), but also unexpected ones. For instance, one configuration did not replace the token in `NewNegotiationItem` (Figure 5) hindering the execution of the next negotiation round. CPN Tools also highlighted another synchronization problem. It may happen when a negotiation fails and the convenience of a new strategy is evaluated. The solution found is based in specific characteristics of the running example and cannot be generalized. Thus, such synchronization issues must be better addressed in future versions of the model.

Simulation. We used CPN tools to run simulations and to produce performance reports. The generated performance report helped us to assess the correctness of the model. We wanted that: (a) new strategies were tried, (2) distinct simulation produced different outputs, (3) all properties reached the final places Agreed, NotAgreed or NotNegotiated (Figure 5), (4) eventually, some auction failed (this guarantees that a property is put in the NotNegotiated place). Figure 24 shows a performance report. Recall Figure 5. Regarding the first concern, a new

strategy is tried when transition `TryNewStrategy` fires. According to the performance report, this happens, in average, 1.74 times in each simulation run (see line `count_iid` of monitor `Count_trans_occur_NegotiateProperty'TryNewStrategy_1`). To assess the second concern, three marking size monitors were assigned to the final places (see columns `Min` and `Max` of monitors `Marking_size_NegotiateProperty'Agreed_1`, `Marking_size_NegotiateProperty'NotAgreed_1` and `Marking_size_NegotiateProperty'NotNegotiated_1`). Note that the maximum numbers of tokens observed in each place is greater than zero. This means that eventually a token reached such places. Note also that the minimum number of tokens differs from the maximum. It means that the outputs can vary in each simulation. To assess the third concern, we grouped the final places and assigned a monitor (Final_places) that counted the total number of tokens in these places. For the running example, the number of tokens in the final places at the end of a simulation must be ever 6. Note that the measures `Min` and `Max` for `max_iid` are both 6. This does not guarantee the correctness of the model, however the opposite would show that the model was incorrect. Finally, a transition occurrence monitor was assigned to `np2` transition (Figure 5) in order to verify the fourth concern. If an auction fails, this transition fires. The performance report showed that this transition eventually fires.

Number of replications: 100

Statistics							
Name	**Avrg**	**90% Half Length**	**95% Half Length**	**99% Half Length**	**StD**	**Min**	**Max**
Count_trans_occur_NegotiateProperty'TryNewStrategy_1							
count_iid	1.740000	0.276611	0.330936	0.439088	1.661325	0	7
max_iid	0.700000	0.076684	0.091745	0.121728	0.460566	0	1
min_iid	0.700000	0.076684	0.091745	0.121728	0.460566	0	1
Count_trans_occur_NegotiateProperty'np2_1							
count_iid	0.080000	0.045398	0.054314	0.072064	0.272660	0	1
max_iid	0.080000	0.045398	0.054314	0.072064	0.272660	0	1
min_iid	0.080000	0.045398	0.054314	0.072064	0.272660	0	1
Final_places							
count_iid	345.070000	10.611371	12.695406	16.844357	63.731960	230	569
max_iid	6.000000	0.000000	0.000000	0.000000	0.000000	6	6
min_iid	0.000000	0.000000	0.000000	0.000000	0.000000	0	0
sum_iid	6.000000	0.000000	0.000000	0.000000	0.000000	6	6
avrg_iid	0.017982	0.000557	0.000667	0.000884	0.003346	0.010545	0.026087
Marking_size_NegotiateProperty'Agreed_1							
count_iid	345.070000	10.611371	12.695406	16.844357	63.731960	230	569
max_iid	5.260000	0.161584	0.193318	0.256496	0.970473	3	6
min_iid	0.000000	0.000000	0.000000	0.000000	0.000000	0	0
Marking_size_NegotiateProperty'NotAgreed_1							
count_iid	345.070000	10.611371	12.695406	16.844357	63.731960	230	569
max_iid	0.580000	0.148130	0.177223	0.235140	0.889671	0	3
min_iid	0.000000	0.000000	0.000000	0.000000	0.000000	0	0
Marking_size_NegotiateProperty'NotNegotiated_1							
count_iid	345.070000	10.611371	12.695406	16.844357	63.731960	230	569
max_iid	0.160000	0.090796	0.108628	0.144128	0.545320	0	2
min_iid	0.000000	0.000000	0.000000	0.000000	0.000000	0	0

Fig. 24. *Performance report*

7 Conclusion

The paper presented the initial steps aiming at the implementation of our negotiation protocol. First, we modeled the core part of this negotiation protocol using CPN Tools. The goal was to simulate a negotiation for the running example presented in Section 3.2. We used the simulation capabilities of CPN Tools to assess the correctness and the adequacy of the model. Although such simulations cannot prove the correctness of the model, they can, at least, show flaws in the model. In fact, the simulations helped us to identify problems that could be repaired. Some weaknesses were also identified and will be taken into consideration in the actual implementation of the model. The simulation also helped us in ascertaining that the protocol is suitable to the purposes it is intended for. Although each negotiator used a simple strategy, the negotiation process provided by/observed during the simulation corresponded to our expectations, and reasonable values were produced.

We also desired to have a stronger confidence about the model's correctness. Thus, we tried to perform state space analysis. We had to simplify the model and even to split it into three distinct sub-models. We managed to obtain full state spaces for the bargain and ballot sub-models, but not for the auction sub-model. Although we did not fully succeed, this process also helped us to identify flaws in the model. We tried to correct them both in the original model and in the sub-models.

Acknowledgment

This paper is partially supported by CAPES/Brazil under grant 4635/06-0, CNPq WebMaps II Project, and FAPESP.

References

1. Weigand, H., Heuvel, W.: Cross-organizational workflow integration using contracts. Decision Support Systems **33**(3) (July 2002) 247–265
2. Krishna, R., Karlapalem, K., Chiu, D.: An ER^{ec} framework for e-contract modeling, enactment and monitoring. Data & Knowledge Engineering **51**(1) (oct 2004) 31–58
3. Angelov, S.: Foundations of B2B Electronic Contracting. PhD thesis, Technische Universiteit Eindhoven (2005)
4. Bacarin, E., Madeira, E., Medeiros, C.: Contract e-negotiation in agricultural supply chains. Intl. Journal of Electronic Commerce (2007) http://www.gvsu.edu/business/ijec (to appear).
5. Jesen, K., Kristensen, L., Wells, L.: Coloured petri nets and cpn tools for modelling and validation of concurrent systems. Intl. J. on Software Tools for Technology Transfer **9**(3-4) (2007) 213–254
6. Bartolini, C., Preist, C., Jennings, N.: A software framework for automated negotiation. In: SELMAS. (2004) 213–235
7. Chiu, D., Cheung, S., Hung, P., Chiu, S., Chung, A.: Developing e-negotiation support with a meta-modeling approach in a web services environment. Decision Support Systems **40**(1) (July 2005) 51–69
8. Reeves, D., Wellman, M., Grosof, B.: Automated negotiation from declarative contract descriptions. In: Proc. of the 5th International Conference on Autonomous Agents, Canada, ACM Press (2001) 51–58
9. Henderson, P., Crouch, S., Walters, R., Ni, Q.: Comparison of some negotiation algorithms using a tournament-based approach. In: Agent Technologies, Infrastructure, Tools and Applications for E-Services. Volume 2592 of Lecture Notes in Artificial Intelligence. Springer (Jan 2003) 137–150
10. FIPA: Fipa abstract architecture specification. Available at www.fipa.org (2000)
11. Malucelli, A., Palzer, D., Oliveira, E.: Ontology-based services to help solving the heterogeneity problem in e-commerce negotiations. Electronic Commerce Research and Applications **5**(1) (2006) 29–43
12. Governatori, G., Dumas, M., ter Hofstede, A., Oaks, P.: A formal approach to protocols and strategies for (legal) negotiation. In: ICAIL. (2001) 168–177
13. Jensen, K.: Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Springer-Verlag (1997)

14. Chen, Y., Peng, Y., Finin, T., Labrou, Y., Cost, S.: A negotiation-based multi-agent system for supply chain management. In Working Notes of the Agents '99 Workshop on Agents for Electronic Commerce and Managing the Internet-Enabled Supply Chain., Seattle, WA, April 1999. (1999)
15. Liu, L., J, B.: Enhancing the CES Protocol and its Verification. Sixth Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools (2005)
16. Dang, J., Huhns, M.: Concurrent Multiple-Issue Negotiation for Internet-Based Services. IEEE Internet Computing **10**(6) (2006) 42–49