A Collaborative Model for Agricultural Supply Chains*

Evandro Bacarin[1], Claudia B. Medeiros[2], and Edmundo Madeira[2]

[1] Departament of Computer Science - UEL - CP 6001
86051-990 Londrina PR Brazil
bacarin@uel.br

[2] Institute of Computing - UNICAMP - CP 6176
13081-970 Campinas SP Brazil
{cmbm,edmundo}@ic.unicamp.br

Abstract. This paper presents a collaborative model for agricultural supply chains that supports negotiation, renegotiation, coordination and documentation mechanisms, adapted to situations found in this kind of supply chain – such as return flows and composite regulations. This model comprises basic building blocks and elements to support a chain's dynamic execution. The model is supported by an architecture where chain elements are mapped to Web Services and their dynamics to service orchestration. Model and architecture are motivated by a real case study, for dairy supply chains.

1 Introduction

A supply chain is a network of retailers, distributors, transporters, storage facilities and suppliers that participate in the sale, delivery and production of a particular product [1,2]. It is composed of distributed, heterogeneous and autonomous elements, whose relationships are dynamic, and change while the chain is activated. Supply chains present several research challenges, such as recording and tracking B2B and e-commerce transactions, designing appropriate negotiation protocols, providing cooperative work environments among enterprises, or coordinating loosely coupled business processes [3].

This paper is concerned with modeling, supervising and coordinating processes in agricultural supply chains, a specific kind of chain that has a large economic impact all over the world. These chains present new challenges in their specification and management, which so far have been mostly ignored by Computer Science researchers.

To start with, the flow within a chain is subject to a wide range of controls. Besides the economic and delivery schedule limitations found in B2B negotiations, agricultural supply chains are sensitive to geographic location, season, climate and product perishability. Examples of concerns are, for instance, whether

* The research reported in this paper was partially financed by CNPq (ORION project), the PRONEX/FINEP/CNPq SAI project and CNPq WEBMaps and AgroFlow projects.

the production process is harmful to the environment or whether it uses genetically modified substances. This requires setting up strict monitoring at all stages, as well as enforcing a large set of rules, which may be product, region or season-sensitive. A parallel concern is the quality of the final product, which involves auditing all production and distribution stages.

Another peculiarity is the so-called "return flow" within such chains, in which the refuse of a given stage of the chain may be recycled and re-enter the chain at another stage. Recycling is not a problem restricted to agricultural chains, but the constraints imposed on these cycles are. Finally, the number and kinds of actors encountered allow limitless possibilities of chain configurations, and the same kind of raw material may originate a large set of interrelated chains.

Our solution combines research in databases, computer networks, and distributed systems and is based on tackling the problem in several stages and levels. The first stage involves *modeling the chain's components* and *dynamics*. Subsequent stages consist in *mapping* the chain to our architecture, whose elements are seen as Web Services.

For each of these stages, the chain's elements and flow have to be considered at two levels: within and across enterprises. Furthermore, service coordination also considers two levels: global dynamics, treated by Coordination Plans; and inter-element dynamics, treated via Contracts negotiated between trading partners.

The main contributions are the following: (i) a general model for specification of agricultural supply chains, which takes into consideration cross organizational collaboration aspects; (ii) an architecture for its implementation, which emphasizes coordination and service flow composition issues; (iii) the validation of the model via a real life case study in agriculture, stressing the peculiarities of this kind of application domain.

The rest of this paper is organized as follows. Section 2 provides an example that will be used throughout the paper to illustrate our work. Section 3 describes the model. Section 4 specifies the architecture and shows how it supports dynamic behaviour. Section 5 outlines an implementation of a chain via Web services. Section 6 contains related work and section 7 concludes the paper.

2 Agricultural Supply Chains

This section presents a simple agriculture supply chain that will be used throughout the paper to illustrate our solution. Figure 1 shows this example – the *dairy cattle supply chain*. The goal of this dairy chain is to process milk, producing and commercializing its products – such as bottled milk, butter, or cheese. The starting point is a "Milk Producer" – a farm that has milk-cows. The farmer gathers milk at given periods. Next, milk is delivered by some sort of transportation means "Transport 1" to a Dairy (production). It can only be processed if it obeys certain constraints stated in "Regulation 1". At the "Dairy", it is processed to create products, which are then transported for wholesale and finally retail commercialization, reaching the end consumer. Products and inputs may be stored at different storage facilities throughout the chain – e.g., warehouses. At each stage, various actors – humans or software – may intervene: lawyers, commodity brokers, quality certifiers or software agents.

Some of the chain's refuse may provide feedback to it, in terms of return flows – such as from the Dairy back to the Producer. For instance, milk that overflows from vats returns to the farms to be used in cattle feed.

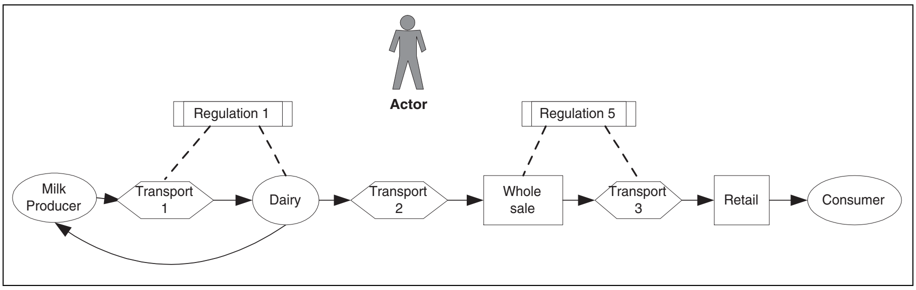

Fig. 1. The Dairy Supply Chain

Even though the diagram in Fig. 1 shows a sequential execution, this is seldom the case. Each chain component may moreover encapsulate other chains. Negotiation, cooperation and coordination issues occur at all levels. Coordination may be centralized – such as in the milk cooperative – or distributed among several coordination centers, that negotiate with each other.

3 A Model for Supply Chains

3.1 Basic Elements

The model's basic elements are Actors, Production, Storage and Transportation. Chain dynamics are furthermore supported by elements Regulation, Contract, Coordination Plan and Summary.

A *Production Element* encapsulates a productive process that uses raw material extracted from its own environment or inputs obtained from other components and produces a product that is passed on to the chain. It is represented graphically by an ellipsis.

A *Storage Element* stores products or raw material and a *Transportation Element* moves products and raw material between production and storage components. They are represented by rectangles and diamonds respectively.

Actors are software or human agents that act in the chain. They may be directly or indirectly involved in the execution of activities A *Regulation Certifier* is an actor that is responsible for certifying that activities or products within the chain obey a set of constraints – such as sanitary regulations or quality specifications.

Regulations are sets of rules that regulate a product's evolution within the chain. These rules specify constraints imposed at distinct execution stages, such as government regulations, quality criteria, or conditions determined by a region's social, cultural, economic or even religious context. Regulations may be atomic or complex, containing other regulations within them.

Interactions among chain components are organized by means of *Coordination plans* and negotiated via *Contracts*. A *Coordination plan* is a set of directives that describe a plan to execute the chain. A chain is coordinated by a top level plan, which may furthermore activate other plans. Plans indicate, among others, sequences of chain elements to be activated, and actors responsible for monitoring these sequences. They trigger activity execution, synchronize parallel activities and control the overall product flow.

Contracts are statements of shared purpose which comprise the mutual obligations and authorizations that reflect the agreements between trading partners [4] that define quality, delivery schedule and costs.

Summaries are elements introduced for traceability and auditability. They are similar to logs, recording chain execution, and may be of two kinds: process and product summaries. A *process summary* contains information about the execution of a production process. A *product summary* stores information on how, when and where a product went through each chain step. It also includes information on certification "stamps" received throughout chain execution.

Dynamics and execution depend on coordination plans, which specify valid element interactions in a very high level. During execution of a specific chain instance, elements are instantiated, contracts negotiated, and the Coordination plan is refined. A Coordination Plan is completely specified only at the end of the execution of a chain, since real-time contract negotiations will dynamically change the chain's configuration, as well as the partners involved.

3.2 Element Composition and Encapsulation

Production, Storage and Transportation elements can be simple or complex. Complex elements are those that can be decomposed into other elements. A complex Production element must include other productive processes, while Transportation and Storage elements cannot encapsulate production elements.

The degree of composition of the elements depends on the level of detail desired. Figure 2 shows how the Dairy Production element of Fig. 1 can encapsulate other production chains. Composition and encapsulation of other elements can be likewise exemplified. Raw milk that arrives at the dairy is pasteurized and stored at the "Milk Warehouse". It may subsequently be bottled within the "Bottling of milk" production element, or be transported via the "Transport 6" element to the "Cheese Production" element.

The placement of a Regulation element within a chain indicates when and where it is applied. "Regulation 1" represents conditions established by the Dairy to accept Raw Milk. They include parameters such as: milk acidity or fat content as well as milk region provenance - e.g., for sanitary reasons. Thus, it is location-sensitive. "Regulation 2" defines rules that determine whether the milk is suitable

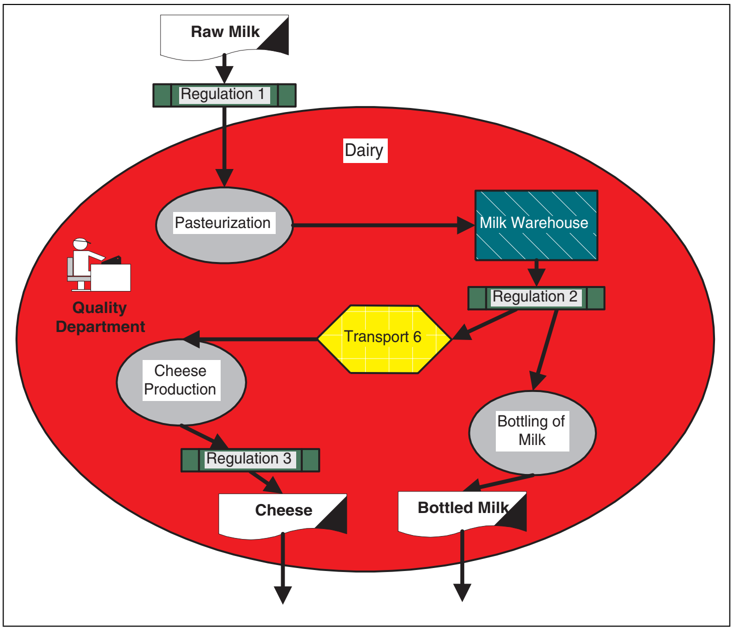

Fig. 2. Breaking down Dairy production element

for cheese or bottling and "Regulation 3" represents quality conditions for cheese comercialization.

Actor "Quality Department" is a sector within the Dairy to check some of the conditions expressed within Regulations 2 and 3. It is within the Dairy element denoting that it can only enforce regulations within it.

3.3 Return Flows

Most supply chain studies ignore return flows, unless they model products returned by a consumer. Waste reuse is seldom considered. Environmental concerns are forcing producers to consider residues. Thus, harmful waste is now being returned to its producer or reprocessed, creating *return flows* in the supply chain. Return flow constraints are modeled within regulations and the flow is modeled by backward or forward links between a chain's components.

4 The Architecture

4.1 Building Blocks

The architecture supports the model described in section 3. It is composed of blocks that encapsulate data and/or services. These blocks can be classified into: those that represent the model's basic elements; those used to support coordination, negotiation, documentation and regulation enforcement; and those used for data needed for a chain's execution and auditing.

The basic elements of our model are directly mapped to the architecture's blocks Production, Storage, Transport and Actor. Manager (M) blocks are introduced in order to handle chain dynamics. The architecture has managers for: coordination (CM), negotiation (NM), regulations (RM) and summaries(SM), that respectively handle coordination plans, contract settlement, regulations and summaries, all mentioned in section 3. Furthermore, distinct kinds of repositories are needed to store information on: chain Participants (the basic elements), Products, Regulations, Contracts and Summaries. The contents and roles of Production, Transport, Storage and Actor blocks are straightforward. There follows a description of manager and repository blocks.

Repository Blocks

Information about chains' elements and execution is stored in six kinds of repositories. Any implementation of the architecture requires that there be at least one repository of each kind, under the responsibility of specific managers.

A *Participant repository* stores cadastral data on a chain's basic participants, namely: Transportation, Storage, Production and Actors. Its goal is to allow validation of the identity of the agents acting within the chain, as well as the roles played by them. It also helps the process of chain instantiation, by supporting the selection of actual businesses to play a given role within a chain.

A *Product repository* contains data on all products and materials used within a supply chain. Its goal is to allow verification of product properties, as well as supporting cross-references within and across chains.

A *Regulation repository* stores regulations for contract negotiation and quality control. Such regulations include global rules (e.g., government level) and local rules (e.g. within a production process).

A *Contract repository* stores contracts established among chain components. More details on these contracts are provided in the next section. A *Coordination plan repository* contains coordination plans specified at distinct granularity levels. The coordination plan repository also contains information about plan execution (e.g., instantiation, validity).

All these repositories support composition of their elements. Thus, composite contracts can be built by aggregating other contracts, plans can be built from the composition of previously stored plans, and so on. Summaries, on the other hand, record the execution of a chain and thus cannot be created from past summaries.

A *Summary Repository* stores product and process summaries, for documentation and auditing. Thus, they can be controlled by government agencies, such as health or sanitation departments, to check on the quality of products and of the production process.

Manager Blocks

The chain's elements and flow have to be considered at two levels: within and across enterprises. Furthermore, service coordination also considers two levels: global dynamics, treated by a Coordination Plan; and inter-element dynamics, treated via the negotiation of Contracts between trading partners. Cooperation, collaboration and negotiation within a chain and the documentation of its activities are handled by manager blocks. Managers may be totally automated or require human Actor intervention.

A *Coordination manager* is in charge of Coordination Plans, interpreting, controling and coordinating them. It is also responsible for managing the Coordination plan Repository. Therefore, these managers trigger and coordinate all processes within the chain. In particular, they are responsible for starting negotiation among components, and may also start regulation enforcement procedures.

A *Negotiation manager* is responsible for handling contracts and coordinating negotiation among distinct chain elements. It also controls Contract Repositories.

A *Summary manager* controls access to a Summary Repository. A *Regulation Manager* encapsulates the access to a Regulation Repository and is also used to verify regulations using information from all repositories. It informs to the Coordination and Negotiation managers whether a regulation has been obeyed or not. Thus, it does not play an active role in regulation enforcement.

4.2 Orchestration of the Supply Chain

The backbone of all orchestration interactions within a chain is formed by a hierarchy of Coordination Managers, that communicate along specific protocols based on a coordination plan. A coordination manager CM at a given hierachical level can only communicate with its parent and its children (levels immediately above and below).

All other interactions among managers are described in terms of this coordination hierarchy background. Each coordination manager CM in the hierarchy may be associated with at most one regulation manager RM, one summary manager SM and one negotiation manager NM. These three managers (RM, SM and NM) are said to be *within the scope* of that coordination manager.

A coordination manager, furthermore, interacts with: the negotiation and summary managers within its scope; and with all regulation managers above its level, and the regulation manager within its scope.

Consider again the "Milk Producer" and "Dairy" elements of figure 1. Suppose that the milk producer is, in fact, a cooperative that agregates several milk farms and the dairy is composed of three production units (for butter, bottled milk and cheese). Figure 3 depicts the block arrangement for those elements and some of their interactions. This example details only production elements, but similar arrangements may also be done for transportation or storage elements. The figure shows a 2-level hierarchy, rooted at CM3.

NM1, RM1 and SM1 are within the scope of CM1 (the cooperative's coordination manager). CM1 can communicate with CM3 (its parent in the coordination hierarchy), with the farms (its children), NM1, RM1 and SM1 (the managers within its scope).

A Negotiation Manager can interact with any other negotiation manager, and with regulation managers of the same scope or above. Negotiation is always triggered by a coordination manager interacting with a negotiation manager.

A Regulation Manager may interact with regulation managers at any level above it. They may also respond to requests from any negotiation manager within the same scope or below its level, and to the coordination manager within the same scope or below its level.

Summary Managers only interact with coordination managers and with any other SM.

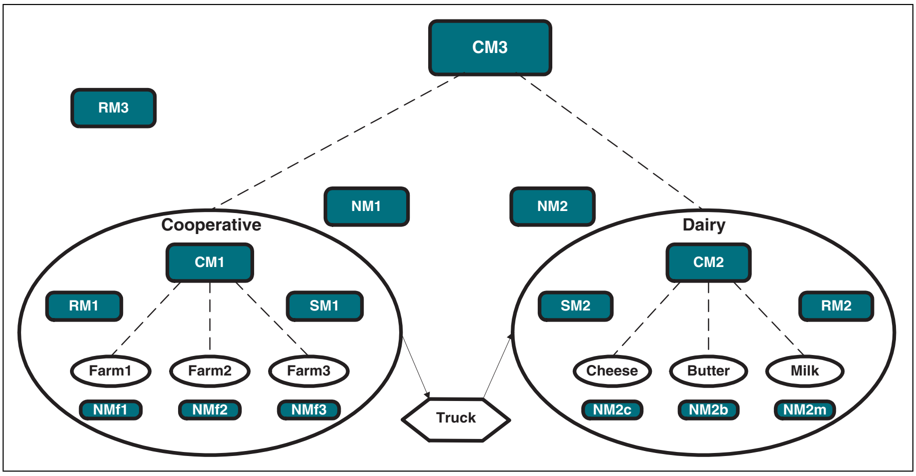

Fig. 3. Illustrating scope and manager hierarchies

4.3 Revisiting the Case Study Using the Architecture

This section illustrates how chain dynamics are supported within the architecture. It starts by discussing coordination aspects, followed by negotiation aspects.

Coordination

The first step in chain execution is its instantiation – this means that a plan's components are instantiated – e.g., Farm 3, registered in the Participant Repository, is a specific farm in Fig. 3. Farms 1, 2 and 3 are furthermore production elements. Each farm has its own negotiation manager (NMf1, NMf2, NMf3). Once the elements start being instantiated, they can agree to establish collaboration, according to coordination plans written and ran by a coordination manager (e.g., CM3). This begins chain execution, started by some coordination manager "higher-up" in the manager hierarchy (CM3) or by actor intervention.

The cooperative and the dairy may undergo several negotiation processes. Those negotiation processes are led by negotiation managers NM1 (for the cooperative) and NM2 (for the dairy). Negotiation is triggered by CM3. In this example, the cooperative and the dairy have their own regulation managers, namely RM1 and RM2. RM3 is responsible for handling regulations within the scope of CM3, and external to the scope of the dairy and cooperative.

Suppose now that CM3, as part of its plan, asks the cooperative to supply 5000 liters of milk the next day. None of the farms can singly afford that volume. Thus, CM1 coordinates this production. It may demand 1000 liters of one farm; 1500 liters of another; and 2500 liters of the last one. As soon as a farm gets the request ready, it reports to CM1. When all farms have reported, CM1 reports to CM3. This kind of communication and execution protocol is similar to that found in management of nested complex transactions in distributed systems [5].

Now, CM3 will ask some transportation agent (Truck) to collect the milk at the cooperative and deliver it to the dairy. When the milk arrives, Truck will notify CM3. CM3 will then ask the dairy to produce 100 liters of bottled milk, 50Kg of butter and 200Kg of cheese. CM2 takes care of this assignment, by coordinating the activities of butter, cheese and bottled milk units. Each unit reports the completion of its task to CM2. When all units have accomplished their tasks, CM2 reports to CM3, and so on.

In this scenario, CM1 and CM2 are subordinated to CM3, but they can coordinate plans that do not depend on CM3, for instance, related to their internal activities.

Negotiation

The relationships among cooperative, farms, dairy (and the respective production units) is governed by contracts. The establishment of a contract is started by a coordination manager that requests intervention from negotiation managers. Consider, again, that CM3 asks the cooperative for a daily production of 5000 liters of milk for the next three months to be delivered to the dairy, but there is not any predefined quota for each farm. The negotiation happens at two distinct levels: the cooperative negotiates with the dairy through NM1 and NM2; the farms negotiate among themselves through NMf1, NMf2 and NMf3.

The negotiation sequence covering both negotiation levels is depicted in Fig. 4. First, CM3 asks the cooperative to deploy a contract negotiation with the dairy. This figure shows that, as soon as CM1 receives a negotiation request from CM3 (edge 1), CM1 starts two activities: a) It asks NM1 (edge 2) to negotiate the contract with the dairy's NM (NM2); b) It asks (edge 3) Farm 1's CM (CMf1) to start milk quotas negotiation among the farms.

As a consequence of CM1's request, NM1 and NM2 develop a negotiation process. NM1 proposes contract clauses to NM2 (edge 4). The latter considers each clause individually and may accept it, reject it or propose an alternative (edge 5). The cycle proposal X alternative runs until they agree to or reject the clause. Eventually, NM1 and NM2 agree to the contract. At the same time, CMf1 asks NMf1 (edge 6) to begin quota negotiation with NMf2 and NMf3 (edges 7 and 8, 9 and 10).

When quota negotiation is finished, NMf1 reports this to CMf1 (edge 11), which in turn relays this information to CM1 (edge 12). Eventually, NM1 and NM2 agree on the deployment contract and NM1 reports the agreement to CM1 (edge 13). As soon as both negotiation processes (milk quotas and deployment contract) are finished, CM1 reports to CM3 (edge 14). Note that eventually NM1 might ask the Cooperative or NM2 might ask the Dairy about some negotiation parameters during a negotiation process. This kind of request is not depicted in this figure.

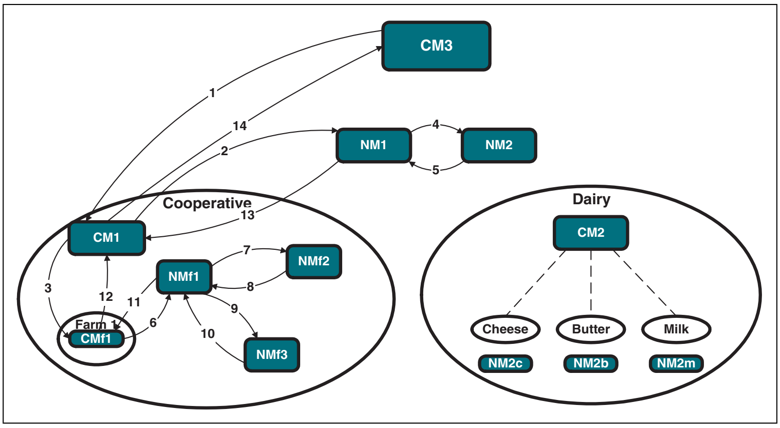

Fig. 4. Coordination and negotiation relationship

A contract is executed and renegotiated on a clause-by-clause basis by the initiative of a coordination manager. For instance, the supply chain may have to be dynamically reconfigured due to a new factor (e.g. a new law, some natural disaster or animal epidemics in a region). Considering the managers illustrated in the Fig. 4, CM3 asks CM1 and CM2 to negotiate new parameters via the suitable negotiation managers NM1 and NM2. These, in turn, verify their contracts in order to determine which contracts and which clauses were affected. The affected clauses are renegotiated individually, again under the proposal X alternative cycle. Negotiation and renegotiation may need human intervention. Each new contract is stored in a Contract Repository by some negotiation manager.

Documentation

The chain execution is documented into summaries that follow products along the chain. Summaries are in fact composed of sequences of local process and product summaries. They are updated at each chain step, and can be merged or subdivided.

Documentation proceeds along the chain. For instance, when the butter unit starts, a new process summary is created for its production process. At the end of this process, the butter unit's CM asks its summary manager to create a summary for the butter produced. This new butter summary is composed of a description of the butter fabrication process, appended to the input milk summary. Eventually, the dairy will output the butter to the next chain step, and this butter will be accompanied by its summary.

Regulation Upholding
Coordination and negotiation involve regulation checking and upholding. For instance, at the end of butter production, CM2 may ask its regulation manager RM2 to check if the product satisfies the suitable restrictions. In order to do this, it will inform RM2 which constraints must be checked. Next, RM2 will combine information from Participant and Product repositories, plus data from the product summary to check these regulations, and return a verdict on regulation compliance, which is also stored in the summary.

5 Implementation

5.1 Mapping into Classes

Implementation of our architecture can be specified in terms of classes in an object-oriented system. Figure 5 uses UML and shows a high level specification of some of the topmost classes needed. The basic elements are in grey, managers are in black, repositories are in white. It shows that basic components include Storage, Transportation and Production, and also the possible compositions among them (Actors are not shown). Note the closed arrowheads from Production, Transport and Storage to Element. This indicates that Element generalizes the other classes, whereas black diamonds indicate composition – e.g., a Production element can encapsulate any other element, whereas Transportation and Storage elements cannot contain Production components.

Black arrows indicate responsibility relationships – e.g., a NM handles contracts, or a SM handles summaries. These classes are implemented in Java. The next section presents highlights of these classes.

5.2 Class Specification

CoordinationManager Class. This class implements the CM block of Fig. 5. A Coordination Manager executes coordination plans. A coordination plan is composed by a set of activities. The coordination plan is a XML file that can be mapped to a BPEL4WS script. The values transferred to and from activities are also XML files. Each activity has an identification and may yield a result after completion. These activities include: execution of another coordination plan, execution of a clause of a contract, verification of a regulation, execution of a Web service operation, and execution of local operations. Activities may be executed sequentially or in parallel and may be synchronized by synchronization primitives.

A given plan can have more than one instance executing at the same time. Thus each plan execution has a unique instance identification. Each plan execution may also receive parameters from the environment.

A CM communicates with a CM within its scope (e.g., CM3 and CM1) via interfaces *CoordinationIF* (Fig. 6) and *ActivityReportIF* (Fig. 7). Orchestration is performed through these interfaces. The lower level CM receives the request

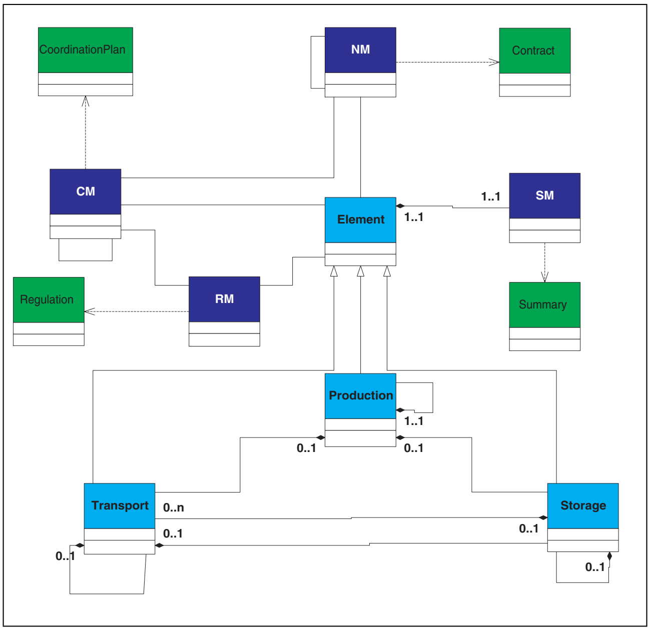

Fig. 5. Class diagram with emphasis in model components and management

```
public interface CoordinationIF {

      public void executeStoredPlan(CoordinationManagerAddress caller,
                                    ActivityIdentification activityId,
                                    PlanIdentification planId,
                                    CoordinationPlanAddress planAddr,
                                    Properties pars);
}
```

Fig. 6. CoordinationIF interface

through its *CoordinationIF* interface and reports the result to the parent's *ActivityReportIF* interface.

Figure 6 shows that the request for plan execution contains parameter *planAddr* that informs the address of the repository where the demanded plan is stored, *planId* is a key that identifies the plan inside the repository, *pars* are environmental parameters, *caller* is the address of the higher Coordination Manager, and *activityId* keeps both the activity and the instance identification of the higher activity that demanded the plan execution. The parameters *caller* and *activityId* are used to report the execution status to the higher manager.

Eventually, the lower manager reports the execution status to the parent manager. Using the received *caller* parameter, it can reach the higher manager and execute *reportPlanStatus* operation of the higher manager (Fig. 7). The parameter *st* informs the status (DONE, ACTIVE, SUSPENDED, RESUMED, CANCELED) and may convey some value produced by the plan's execution, and *activityId* received previously is assigned to *id*.

```
public interface ActivityReportIF {
  public void reportPlanStatus(ActivityIdentification id, PlanStatus st);
}
```

Fig. 7. ActivityReportIF interface

The Coordination Manager has another interface called *OwnerComponentIF* that is quite similar to *CoordinationIF* interface. This new interface is used by a component to demand an inner Coordination Manager the execution of a plan. The execution may be synchronous or asynchronous, and there is an operation to ask the status of an asynchronous plan execution.

ActivityReportIF interface also receives reports from other kind of activities in a similar way.

RegulationManager Class. This class implements the RM block of Fig. 5. An instance of this class verifies regulations. A regulation is evaluated against a summary of a product to verify if that product satisfies the constraints expressed in the regulation.

A Regulation is specified in an XML file (Fig. 8). It contains a section (tag *verify*) with the conditions that must hold for the regulation to be satisfied (the regulation is said to be satisfied). The evaluation of this condition may produce a certificate stamp - another XML file.

```
<regulation id=‘‘unique_id’’ type=‘‘CategoryName’’>
      <parameters> ... < /parameters>
      <enforce>
         <reg var=‘‘VarName’’ id=‘‘RegulationIdentification’’
            address=‘‘RegulationRepositoryAddress’’ >
            <par name=‘‘Parameter1Name’’> Parameter1Value< /par>
         < /reg>
      < /enforce>
      <verify> < /verify>
      <action>
         <ifok>
            <mark m=‘‘all|alltrue|allfalse|#VarName|##’’/ >
         < /ifok>
     < /action>
< /regulation>
```

Fig. 8. Regulation XML file

Complex Regulations embed other regulations to be verified (tag *enforce*). The value produced by the evaluation of an enforced regulation is assigned to *VarName*. A complex regulation is satisfied iff its condition holds and so do all the enforced regulations.

The *action* tag indicates whether to store the certification stamps in the summary or not. The *mark* tag will instruct which mark is appended to the summary; e.g, *all* means that all stamps will be appended; *alltrue* appends the stamps whose value is yes; *allfalse* is the opposite; *#VarName*, appends the stamp contained in variable *VarName*; *##*, appends only the stamp produced by the composite regulation.

5.3 Implementation as Web Services

All architecture elements can be seen as implemented through or encapsulated by Web Services. The only exception is the coordination plan, which is mapped to a workflow.

In more detail, repositories and contracts are static entities encapsulated by Services that provide access to them. Actors can be either Services (e.g., a broker) or Service clients. All managers correspond to services, and the remaining architecture elements – Production, Transportation and Storage – are atomic services or the result of service composition via coordination plans.

The workflow that describes a coordination plan is constructed just as any workflow described in the literature [6], i.e.:

– totally predefined before execution; or
– constructed in an *ad hoc* manner by the CM responsible for the orchestration, while the chain is executed, typical of scientific workflows (e.g., [7,8]); or
– a combination of both.

Each workflow activity references a service responsible for its execution. For instance, in figure 3, a coordination plan executed by CM2 is a workflow that contains an activity that starts cheese production. This activity must refer to the cheese unit (a Service), the desired kind of cheese (a Service for a Product Repository) and the regulations (a Service to a Regulation Repository) that must be verified during the production process in order to ensure cheese quality.

There follows the ennumeration of the interfaces of these Services, which can also be depicted as WSDL specification. Most of these blocks also implement an administration interface, used to configure the corresponding Web Service. The main interfaces of Transportation, Production and Storage Services are:

– *Interfaces for specific/business services:* each element represents a chain partner (e.g., business, enterprise, industry), and therefore can have one or more interfaces for its specific services.
– *Contract Negotiation interface:* receives requests from the Negotiation Manager about negotiation and contract parameters.
– *Contract Execution interface:* accepts requests from other components (or Coordination Manager) to execute a specific contract clause.

– *Sumary Management interface:* responsible for exchange and certification of summaries, via communication with the Summary Manager.

A Coordination Manager Service implements at least the following interfaces. The Java specifications of some of them are shown in section 5.2:

– *Coordination interface:* receives requests from a higher Coordination Managers. Orchestration happens through this interface.
– *Activity Report interface:* receives status reports about the activities demanded from another Service.
– *Owner Component interface:* the interface by which a Coordination Manager receives requests from the component that owns it.

The interfaces implemented by a Negotiation Manager Service include:

– *Negotiation Coordination interface:* accepts requests from the Coordination Manager.
– *Peer Negotiation interface:* for negotiation with another Negotiation Manager Service.

A Summary Manager Service has one *Exchange interface* for exchange of summaries among summary managers.

Finally, a Regulation Manager Service has one interface *Regulation Verifying interface*. It is responsible for checking all rules within a regulation against the chain's state. This may require requesting information from all repositories. It may be invoked by one or more chain components. The component that invoked it is responsible for enforcing the corresponding regulation.

All repositories are encapsulated by Services. The interfaces of these Services offer access to these data for retrieval and update. These interfaces can be accessed by the Managers of a chain and also by external services and systems that have no connection with a chain, but want to perform queries on products, participants, contracts and plans.

6 Related Work

There are several issues that can be analyzed under the umbrella of supply chains - e.g., concerning algorithms adopted, logistics, placement strategies, partner choice. One particular trend, called by [2] IT-related supply chains, concerns information technology tools and techniques to specify and implement such chains. In particular, a recent direction concerns the communication technologies adopted. Problems encountered in electronic commerce and B2B applications and interactions are the same as those faced by supply chain interactions [9].

Though there are many proposals for combining workflows and Web Services (e.g., [10] on agriculture) proposals for supply chains combining these mechanisms are still preliminary. The closest is the research on e-business using Web services, but for other goals – e.g., see [11]. [12] even states that the main reason for the lack of practical implementation of strategic supply chain development can be found in the high degree of complexity that is connected with the identification of supply chain entities and the modelling of the chain structure, as well as the high coordination effort.

Our goal is to contribute to solving these issues. Most researchers do not examine the entire chain, focusing only on some aspects. Auditing structures and log maintenance are ignored. Agricultural chains are mostly examined under a business or logistics framework.

Examples of such approaches are the work of [13] or [14]. The first categorizes integrated supply chains into three models, namely: channel master, chain web, and chain organism. The author states that the predominant model in agricultural supply chain is the channel master. In this model, a dominant firm specifies the terms of trade across the entire supply chain and the coordinated behaviour is based on specification contracts. [14] discusses the usage of information technology in the american cattle-beef supply chain. The paper emphasizes the need for better information integration and well-defined means for describing and enforcing activitities coordination, negotiation and execution of contracts.

Since our proposal is based on Web services implementation, we also examine a few related issues. Two aspects have to be considered: mapping a chain's components to Web services and composition of these services.

[15] analyzes issues in service composition and comments on various standards for orchestration and choreography, such as BPEL4WS, WSCI and BPML. Important concerns in service execution in this context are long running transactions and exception handling. The actions in those standards are undone by compensation actions. This affects documentation of chain execution, since all performed actions are logged in summaries and in repositories. [16], in turn, overviews several proto-patterns for architecting and managing composite Web services, while [17] is more concerned with service semantics.

[18] proposes a mechanism for service definition and coordination. Their architecture is based on a 2-level workflow. At the highest level, a workflow orchestrator controls execution, while at the lowest level service execution can be controlled by a regular workflow engine. This is done through entry points placed between activities. In contrast, the work of [19] uses statecharts for defining service composition, and is based on a distributed orchestration engine.

[20] proposes a service-oriented architecture built upon the Web services proposals for inter-enterprise and cross-enterprise integration. Using this architecture, process managers can compose and choreograph business processes based on exposed enterprise and Web services.

Several other authors are concerned with organizational and modeling aspects of supply chains, as indicated by the classification proposed by [2] to analyze efforts in supply chain modeling. This includes for instance work on partner coordination [1], logistics [21] or business contract languages [4].

7 Conclusions

This paper presented a framework for modeling, supervising and coordinating processes in agricultural supply chains. This framework is comprised of two parts: (i) a model for these production chains, that covers both declarative and dynamic aspects; and (ii) an architecture to support the model, based on Web Services and their interfaces.

The model takes into account the fact that agricultural chains are inherently heterogeneous, and sensitive to different kinds of constraints. Chain definition using this model involves specifying its basic components (Actors, Transportation, Process and Storage) and the components needed for cooperation, collaboration, negotiation and documentation (Contracts, Coordination plans, Policies and Summaries). The model provides rules for composition and construction of these elements, thereby allowing *ad hoc* chain construction and execution. The model is mapped into an architecture of Web Services that provides support for contract negotiation, plan coordination, regulation enforcement and summary management. These services also encapsulate access to distinct repositories, that contain data on the chain's partners, processes, policies, constraints, contracts and execution documentation. This architecture supports flow execution at two dimensions: within and across enterprises, for a multiple hierarchy of coordination levels, under service orchestration. Service coordination encompasses global and local dynamics, enforceable by communication protocols established among and across coordination levels.

The main contributions are thus the following: (1) an information technology-based model for specification of agricultural supply chains, which takes into consideration scope, structure and goals, and supports coordination, cooperation and documentation; (2) an architecture for its implementation, which emphasizes negotiation, regulation management, coordination and service flow issues; (3) validation of the model via a real life case study in agriculture.

Current work includes refining the object model of the framework, which will in turn allow implementation and testing of the architecture. This includes testing the suitability of scientific workflows to support the dynamics of ad-hoc coordination plan construction. The implementation will be tested against case studies provided by Brazil's agriculture ministry research corporation.

References

1. Kumar, K.: Technology for supporting supply chain management. Communications of the ACM **44** (2001) 58–61
2. Min, H., Zhou, G.: Supply chain modeling: past, present and future. Computer & Industrial Engineering **43** (2002) 231–249
3. Arsanjani, A.: Developing and Integrating Enterprise Componentes and Services. Communications of the ACM **45** (2002) 31–34
4. Weigand, H., Heuvel, W.: Cross-organizational workflow integration using contracts. Decision Support Systems **33** (2002) 247–265

5. Oszu, T., Valduriez, P.: Principles of Distributed Database Systems. Prentice Hall (1991)
6. Gal, A., Montesi, D.: Inter-enterprise workflow management systems. In: Proc. 10th International Conference and Workshop on Database and Expert Systems Applications (DEXA '99). (1999) 623–627
7. Weske, M., Vossen, G., Medeiros, C.B., Pires, F.: Workflow Management in Geoprocessing Applications. In: Proc. 6th ACM International Symposium Geographic Information Systems – ACMGIS98. (1998) 88–93
8. Cavalcanti, M., Mattoso, M., Campos, M., Llirbat, F., Simon, E.: Sharing Scientific Models in Environmental Applications. In: Proc ACM Symposium Applied Computing - SAC. (2002)
9. Medjahed, B., Benatallah, B., Bouguettaya, A., Ngu, A., Elmagarmid, A.: Business-to-business interactions: issues and enabling technologies. The VLDB Journal **12** (2003) 59–85
10. Fileto, R., Liu, L., Pu, C., Assad, E., Medeiros, C.B.: POESIA: An Ontological Workflow Approach for Composing Web Se rvices in Agriculture. VLDB Journal **12** (2003)
11. Rust, R., Kannan, P.: E-Service: a New Paradigm for Business in the Electronic Environment. Communications of the ACM **46** (2003) 36–42
12. Albani, A., Keiblinge, A., Turowski, K., Winnewisser, C.: Identification and modelling of web services for inter-enterprise collaboration exemplified for the domain of strategic supply chain development. In Meersman, R. e.a., ed.: CoopIS/DOA/ODBASE 2003. (2003) 74–92
13. Peterson, H.: The "learning" supply chain: Pipeline or pipedream? American J. Agr. Econ. **84** (2002) 1329–1336
14. Salin, V.: Information technology and cattle-beef supply chains. American J. Agr. Econ. **82** (2000) 1105–1111
15. Peltz, C.: Web services orchestration: a review of emerging technologies, tools, and standards. Technical report, Hewlett Packard, Co. (2003)
16. Benatallah, B., Dumas, M., Fauvet, M., Rahbi, F., Sheng, Q.: Overview of some patterns for architecting and managing composite web services. ACM SIGecom Exchange **3** (2002) 9–16
17. Bussler, C., Fensel, D., Maedche, A.: A Conceptual Architecture for Semantic Web enabled Web Services. ACM Sigmod Record **31** (2002) 24–30
18. Belhajjame, K., Vargas-Solar, G., Collet, C.: Defining and coordinating open-services using workflows. In et al., R.M., ed.: CoopIS/DOA/ODBASE 2003. (2003) 110–128
19. B. Benatallah, Q.Z. Sheng, M.D.: Environment for web services composition. IEEE Internet Computing (2003) 40–48
20. P. Fremantle, Weerawarana, S., Khalaf, R.: Enterprise Services. Communications of the ACM **45** (2002) 77–82
21. Simon, S.: The art of military logistics – moving to dynamic supply chain. Communications of the ACM **44** (2001) 62–66