Cerere: an information system supporting traceability in the food supply chain

M.G.C.A. Cimino, B. Lazzerini, F. Marcelloni, A. Tomasi
Dipartimento di Ingegneria dell'Informazione: Elettronica, Informatica, Telecomunicazioni
University of Pisa
Via Diotisalvi 2, 56122 Pisa (Italy)
Fax: +39 050 2217600; Tel: +39 050 2217599
{m.cimino, b.lazzerini, f.marcelloni, a.tomasi}@iet.unipi.it

Abstract

In this paper, we present a system for traceability in the food supply chain. The system is able to systematically store information about products and processes operating on products throughout the entire supply chain from farm suppliers to retailers. The system manages quality information too. Thanks to the use of the electronic business using eXtensible Markup Language (ebXML) standard, the traceability system also provides data homogeneity, scalability and interoperability.

1. Introduction

According to the ISO 9001:2000 standard, *chain traceability* is the ability to trace the history, application or location of an entity by means of recorded identifications throughout the entire food chain. In practice, chain traceability is achieved if food businesses keep records of suppliers and customers and exchange this information along the entire food supply chain. In particular, each unit/batch (called *lot* in the following) of an ingredient or a product must be both *traceable* and *trackable*. To trace an entity means to identify its origin by tracing back in the supply chain, whereas to track an entity means to follow the path of the entity through the supply chain from supplier(s) to consumers. Traceability in the food supply chain has attracted considerable attention in the last few years for a variety of reasons. First of all, it has become a legal obligation within the EU from 1st January 2005 [1]; similar requirements for traceability systems are present in the United States and Japan too [2][3]. Then, food companies tend to consider the significant expenditure required to build a traceability system as a long-term strategic investment to create consumer confidence both in the image of the company and in the specific product. As a consequence, other requirements for traceability exist besides the legal ones. In fact, in addition to systematically storing information that must be made available to inspection authorities on demand, a traceability system should take also food safety and quality improvement into account [4]. This means, for example, enabling the system to trace back to find out the cause of a problem and prevent it from happening again, or to launch a proper recall of potentially unsafe products that have already been forwarded, thus protecting public health. Of course, the implementation of a complete and efficient food traceability system has to cope with several problems, such as the lack of alignment of the possibly different systems adopted in the various sectors of the food supply chain, or the non homogeneous information kept at the various units of the supply chain [5][6]. To build a traceability system is therefore a complex task that involves all stages of production, processing and distribution: traceability records should be kept for both products and processes (such as moving, transformation or combination) that operate on products.

In this paper, we describe *Cerere*, a generic traceability system for the food supply chain, which is able to meet both legal and quality requirements. Since the reliability of such a system heavily relies on the reliable and faithful exchange of documents among the various units of the supply chain, we adopt ebXML [7] as a standard that can help support data homogeneity and scalability as well as system interoperability.

2. Traceability

As previously stated, a traceability system must be able to trace both lots and activities [8]. This means that each data model related to traceability must include lot and activity as key entities, and allow lot tracking and tracing. Tracking refers to the ability to follow the downstream path of a product along the supply chain, maybe based on specific criteria. This is a crucial factor, e.g., for an efficient recall of defective products. Tracing, on the other hand, refers to the ability to determine the origin and characteristics of a particular product. This is obtained by referencing to records held upstream in the supply chain. Tracing can help detect the cause of quality problems. Figure 1 shows a simplified supply chain consisting of only four segments: it shows a typical scenario of a product recall due to a contamination event. In the figure, a circle denotes a *traceability lot* (*lot*, for short), that is a unit of the food product processed or packaged under the same conditions, or a batch of products that share such characteristics as type, category, size, package and place of origin. A rectangle represents an *activity*, such as production, packaging, distribution and sale, which may receive *N* lots as input and may deliver *M* lots as output. An edge represents the relation between a lot and an activity. In this way, edges allow following the path of lots along the chain. The supply chain unit, which performs an activity, is responsible for the activity itself and for the corresponding outgoing lots. In the following, the unit is denoted as *responsible actor*. Assuming that each lot is generated by an activity, we can state that each lot is associated with a responsible actor. For traceability purposes, this actor is also responsible for the reliability of the traceability data related to the lot.

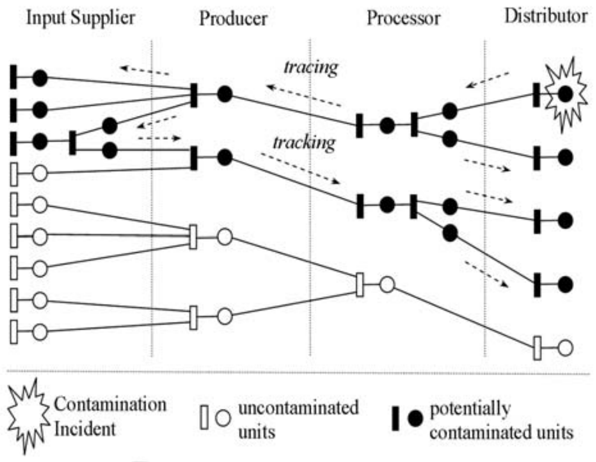

Figure 1. Typical scenario for a product recall in a supply chain.

The presence of an efficient traceability system allows constraining the product withdrawal or recall only to the products really affected by contamination. Tracing and tracking capabilities are therefore crucial to confine the reaction to possible hazards and reduce the recovery cost.

The scenario shown in Figure 1 requires the adoption of an appropriate data model, which must be general enough to represent any kind of food. It also has to provide a means to univocally identify traceability lots and activities, record information about lots and activities, and their relations. It would also be advisable to include additional data regarding, in particular, food quality. For example, when a cooking activity is involved, oven temperature and humidity could be important parameters to help avoiding cases of hazard.

Each lot must be identified by a *global identifier*, which has to be univocal within the supply chain. To avoid a centralized administration of the identifiers, we adopt a solution inspired to the approach used in the EAN/UCC standard. We assume that each actor is uniquely identified in the supply chain by an *actor identifier*. We allow an actor to associate freely an identifier (*traceable entity identifier*) with each traceable entity, that is, either an activity or a lot, the actor is responsible for. If an actor produces more products, the lot identifier may consist, for instance, of the type of product and a progressive number. The only constraint we impose is that the identifier is univocal within the amount of lots managed by the actor. The *global identifier* is composed of the *actor identifier* and the *traceable entity identifier*.

Figure 2 shows the data model using a UML notation. Here, two distinct packages are shown: Traceability and Quality. The former contains the entities that allow tracing and tracking the product path. The latter contains the components related to lot quality. The Traceable Entity is an abstract class which models the basic characteristics of the two entity types involved in traceability: lots and activities. The field TE_ID implements the traceable entity identifier. The association relation *is managed by* enforces a traceable entity to be always associated with a responsible actor. This constraint guarantees the univocal identification of the traceable entity, as described above. Further, Traceable Entity is also associated with Site, which has its own unique identifier. This relation states that each lot is contained within a site. Thus, at each stage of the supply chain, the traceability system is able to retrieve the information about the site where the lot has been processed or stored. Both Site and Responsible Actor are characterized by a number of attributes, which summarize all the information needed for traceability. Classes Lot and Activity are derived from Traceable

Entity. The association relation *is generated from* means that each lot may be generated from one or more lots. The generation is ruled by an activity.

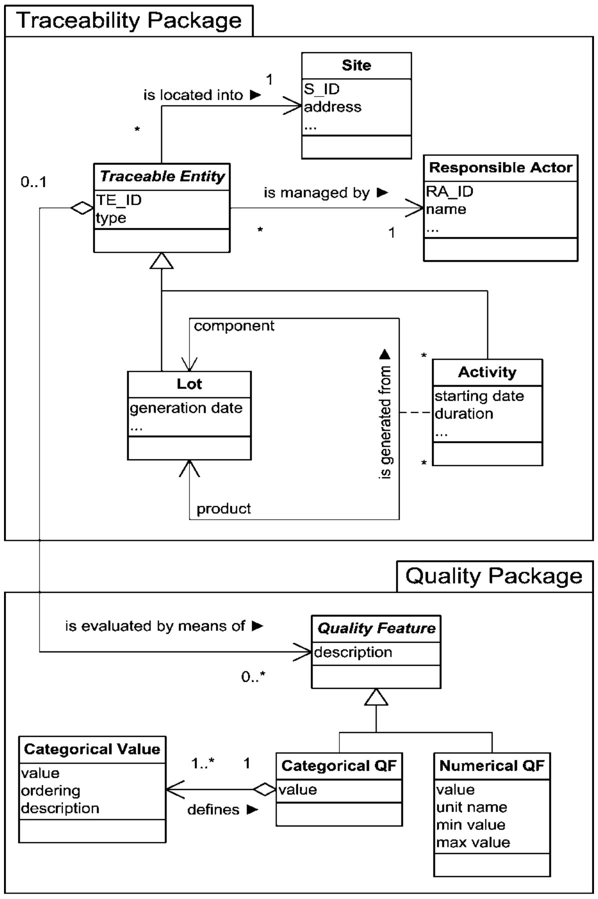

Figure 2. UML class diagram of the traceability data model.

Figure 3 shows an example of the objects used to record an activity: a distributor purchases a red wine cask from a producer, and carries it to her/his storehouse by a truck. The input and the output lots of the activity are definitely the same cask lot. However, producer and distributor typically identify the lot in a different way. Further, producer and distributor are, respectively, responsible for the input and the output lot. Therefore, for traceability purposes, input and output lots are different.

In Figure 4 two UML sequence diagrams describe the interaction of messages arranged in time during a purchase. In the first diagram we adopt a distributed architecture without a central database. Here, the unit responsible for an activity is also responsible for recording and managing the relation between input and output lots. The producer communicates the global identifier of the input lot to the distributor, which provides to associate it with the global identifier of the output lot. This association allows tracing and tracking the lot. Typically, the global identifier is attached as bar code or RFID tag to the lot. Thus, part of the communication between supply chain units generally consists of reading the identifier by appropriate readers.

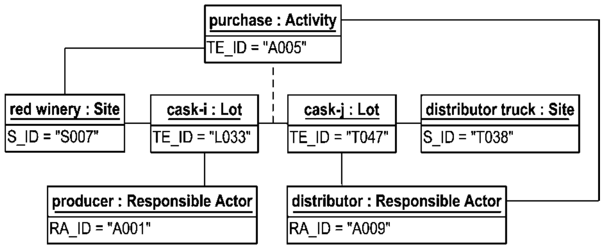

Figure 3. Objects involved in recording the actual execution of a simple activity.

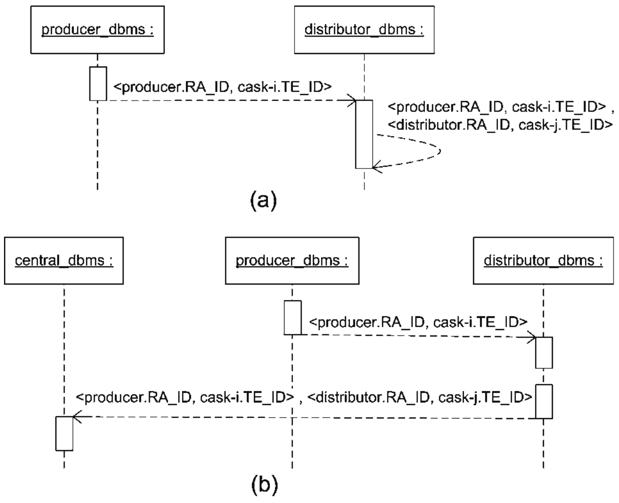
producer_dbms :
distributor_dbms :
<producer.RA_ID, cask-i.TE_ID>
<producer.RA_ID, cask-i.TE_ID> ,
<distributor.RA_ID, cask-j.TE_ID>
(a)
central_dbms :
producer_dbms :
distributor_dbms :
<producer.RA_ID, cask-i.TE_ID>
<producer.RA_ID, cask-i.TE_ID> , <distributor.RA_ID, cask-j.TE_ID>
(b)

Figure 4. Sequence diagram of a purchase activity; (a) distributed architecture (b) centralized architecture.

In the second diagram a central database exists, which is responsible for the traceability data. This architecture requires that each supply chain unit responsible for an activity provides the database with all the information related to the activity. In particular, this information must allow the database at least to associate the input lot(s) with the output lot(s).

In either architectures, in order to retrieve a lot history, the various units in the supply chain have to communicate with each other and possibly with the central database. The data exchange must of course be performed in a secure and reliable way.

Further, quality requirements should also be taken into account. The ISO 9000 standard defines quality as the totality of features and characteristics of a product or service that bear on its ability to satisfy stated or implied needs. To meet quality requirements, we introduced the Quality Package shown in Figure 2. This package contains the abstract class Quality Feature, which includes a description of the feature itself and a collection of methods to set and retrieve feature values. Values can be either categorical or numerical. Categorical QF and Numerical QF concrete classes implement features that can assume, respectively, categorical and numerical values. Categorical QF contains a set of Categorical Value objects, which define the possible values. A Categorical Value is characterized by the value, a description, and an ordering number. This last item can be used whenever ordered categorical values are needed. Numerical QF is qualified by the value, the unit name (for instance, Kg for "weight" quality factor), and the minimum and maximum values. This class organization allows dealing uniformly with different quality features.

Figure 5 shows an example of object diagram that describes the quality features "color intensity" and "rating" associated with lot *wine cask*. Color intensity can assume numerical values in the interval 1,10. Rating takes the wine excellence into account. Here, excellence is evaluated by using three values: one star, two stars, and three stars, which correspond, respectively, to good, very good and excellent.

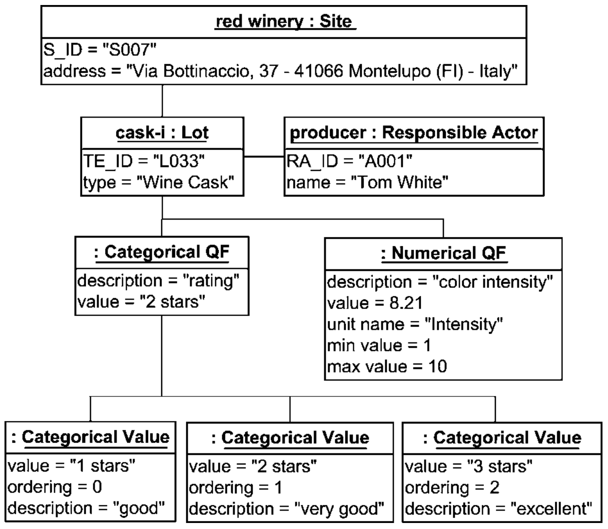

Figure 5. Example of objects related to quality features.

3. Dynamic behavior

The dynamic behavior of the lot can be modeled by the following six activity patterns [3].

1. *Lot acquisition*: an actor (*buyer*) of the supply chain acquires a lot from another actor (*provider*). Since a lot can have only one responsible actor, the buyer generates a new lot and creates an association between the pre-acquisition lot and the post-acquisition lot; this association allows implementing the tracing process and therefore determining the origin and characteristics of a particular product. Figure 6.a shows the lot acquisition pattern using a UML activity diagram, whose aim is to focus on object and control flows driven by internal processing. The shape with straight top and bottom and with convex arcs on the two sides represents an action state, i.e., a state with an entry action and at least one outgoing transition involving the implicit event of completing the entry action. Rectangles represent objects. The input and output objects of an action state are connected to the action state by a dashed arrow. For the sake of simplicity, we have omitted the object Activity as output of the action state. This object maintains the relation between the pre-acquisition lot and the post-acquisition lot.

2. *Lot providing*: an actor (*provider*) of the supply chain provides another actor (*buyer*) with a lot. The provider generates a new lot and creates an association between the pre-providing lot and the post-providing lot; this association allows implementing the tracking process and therefore following the downstream path of a product along the supply chain. Figure 6.b shows the lot providing pattern.

3. *Lot division*: a lot is split into a number of lots. The responsible actor of the lot creates an association between the pre-division lot and the post-division lots, and vice versa. Thus, both tracing and tracking processes are possible. Examples of lot division are cutting and splitting. Figure 6.c shows the lot division pattern. The black bar represents a concurrent transition: the post-division lots are forked concurrently from the pre-division lot.

4. *Lot integration*: a number of lots are integrated into a unique lot. The responsible actor of the lot creates an association between the pre-integration lots and the post-integration lot, and vice versa. Examples of lot integration are mixing and packing. Figure 6.d

shows that pre-integration lots are concurrently integrated into the post-integration lot.

5. *Lot alteration*: as shown in Figure 6.e, a new lot is generated from a lot by an alteration activity. The responsible actor of the lot creates an association between the pre-alteration lot and the post-alteration lot, and vice versa. Examples of lot alteration are heating, freezing and drying.

6. *Lot movement*: a lot is moved from a storage to another storage under the same responsible actor. Since a lot can be associated with a unique site, the responsible actor has to create a new lot with a new identifier. Further, the responsible actor creates an association between the pre-movement lot and the post-movement lot, and vice versa. The movement of a lot from a storage to another storage can be considered as a lot alteration which does not change the lot features, but only the lot site. Thus, lot movement can be represented as in Figure 6.e.

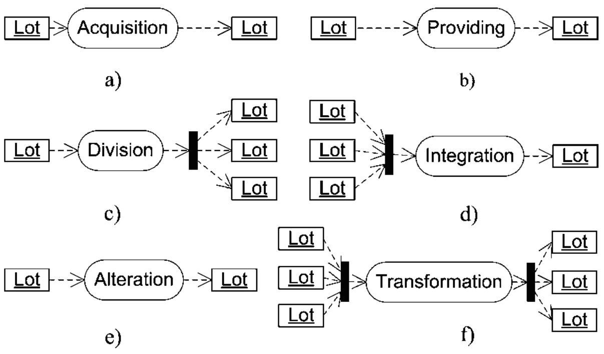

Figure 6. Patterns of the dynamic behavior of the lot.

From a dynamic point of view, lot division, lot integration and lot alteration can be modeled as a generic lot transformation from N lots to M lots (see Fig. 6.f). Thus, a division of a lot into N lots and the integration of N lots into a unique lot are represented as a transformation of one lot into N lots and of N lots into one lot, respectively. Unlike the other patterns, in the first two patterns, the post-activity lots and the pre-activity lots have different responsible actors. A lot has to be acquired before being transformed. But if there exists an actor that acquires a lot, there must also be an actor that provides the lot. Further, a lot, which has been acquired (provided), cannot be acquired (provided) again if it has not been preliminarily provided (acquired). These observations lead us to define the state diagram of a lot as in Figure 7.

The lot can be in one of three different states: *acquired*, *transformed* and *provided*. The figure highlights the constraints intuitively expressed above. First, the state of a lot cannot change directly from "acquired" to "acquired" or from "provided" to "provided". When a lot is acquired, it can be either transformed into another lot (transformation activity) or provided (providing activity) to another actor. Second, when a lot has been provided to an actor, it can be only acquired (acquisition activity) by another actor. On the contrary, the state of a lot can change from "transformed" to "transformed". This occurs, for instance, when the lot undergoes different movements or manufacturing processes. When a lot has been transformed, however, it cannot be acquired directly, but it has to be firstly provided by the responsible actor and then acquired by another responsible actor.

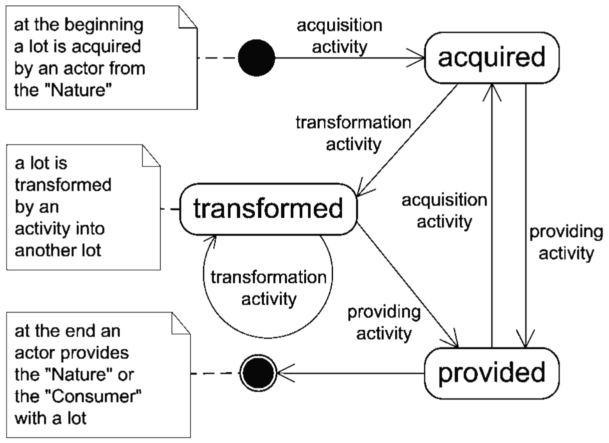
at the beginning a lot is acquired by an actor from the "Nature"
acquisition activity
acquired
transformation activity
a lot is transformed by an activity into another lot
transformed
acquisition activity
providing activity
transformation activity
providing activity
at the end an actor provides the "Nature" or the "Consumer" with a lot
provided

Figure 7. Lot state diagram.

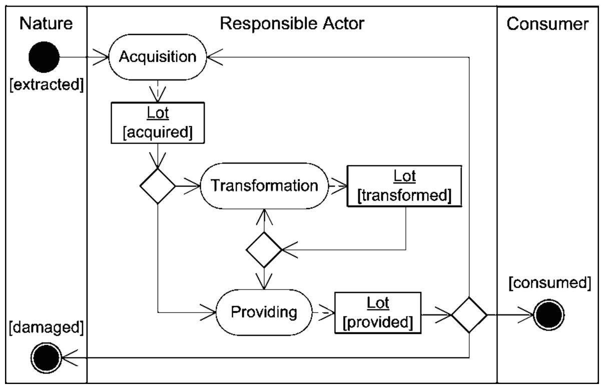

Figure 8. UML activity diagram describing the dynamic behavior of a lot.

Figure 8 shows a UML activity diagram describing the dynamic behaviour of a lot. The rectangles, denoted as swimlane in UML standard, which contain objects and action states, are used to organize responsibility for actions and objects. In the diagram, the black circle and the black circle included in a white circle are, respectively, the initial and final state of the lot. The white diamond represents a decision. The text in square brackets within an object defines the state of the object. The solid lines identify the control flow. The text in square brackets close to a solid line represents a guard condition, i.e., a Boolean expression written in terms of parameters of the triggering event. In the figure, we introduced two particular responsible actors, namely Nature and Consumer, which correspond to the initial and the final responsible actors of a supply chain. Actually, Nature can also be considered as the final responsible actor when a lot is discarded because, for instance, it is damaged. A lot can be acquired from the nature through an extraction or provided by a responsible actor. Once acquired, a lot can be either transformed or provided. A transformed lot can be either transformed once again or provided. A provided lot can be sent to either another responsible actor, or a consumer or the nature (in case it is discarded).

As an example, let us consider the simplified cheese supply chain shown in Figure 9. The starting point of the supply chain is the milk, which is soured with lactic acid bacteria by the supplier. After the thickening of the milk, the produced gelatin is reduced to small pieces with cutting and mixing tools, then it separates into curds (the solid components of the milk) and whey (the water contained in the milk). At this stage of the production process, the producer has to decide whether a soft cheese, a cut cheese or a hard cheese will be made. The type of cheese implies, for instance, the temperature of the curd and its size. The cheese curd is put into forms and pressed based on the cheese type. Then, the cheese is salted to let the rind form. Finally, the cheese is put into special ripening rooms: the ripening process is controlled by the humidity in the air, temperature and maintenance of the cheese surface.

Figure 9 shows the UML collaboration diagram (at instance level) of the simplified cheese supply chain. In the diagram, *Nature, Supplier, Shop* and *Consumer* are objects that play the role of *Responsible Actor* and exchange with each other stimuli representing procedures, possibly producing new lots. The order of execution of the procedures is described by the numbers associated with the procedures themselves.

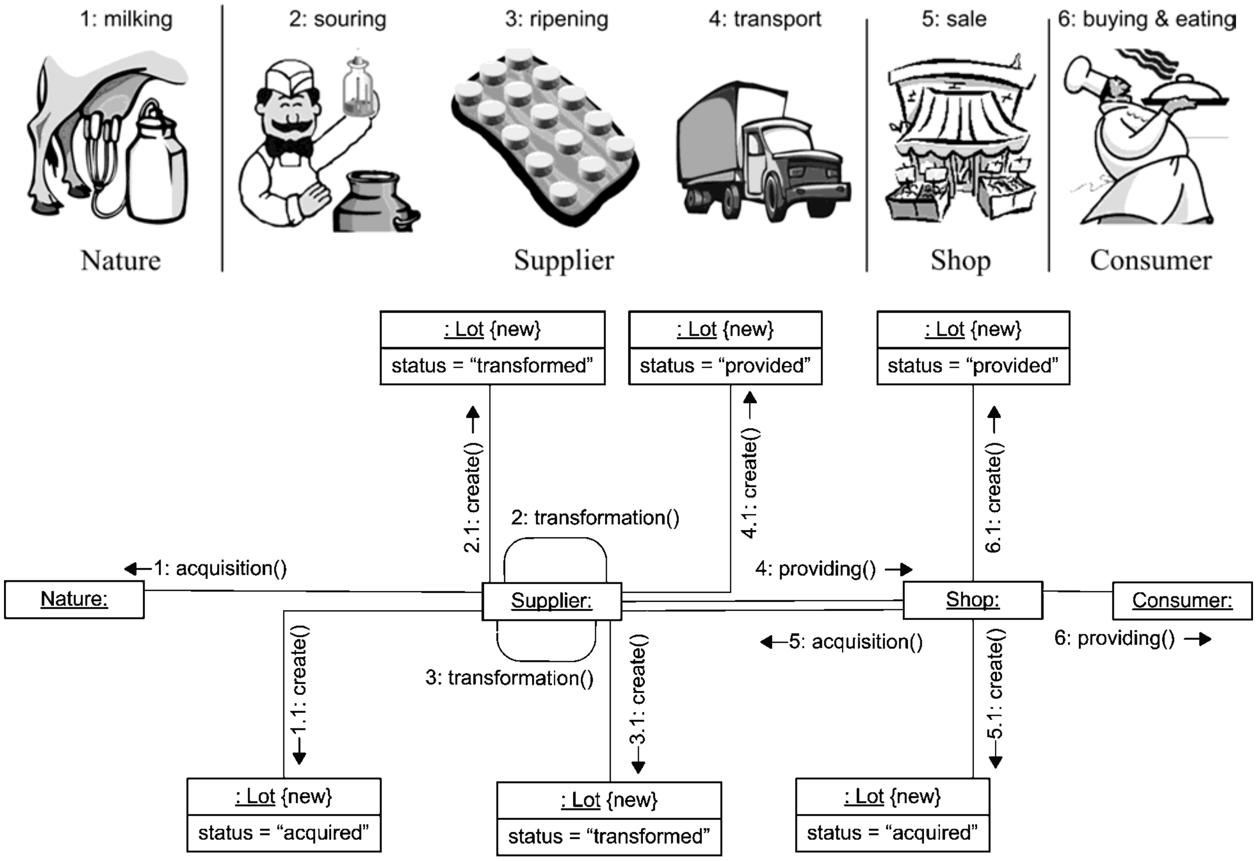

Figure 9. Simplified cheese supply chain.

At the beginning, the *Supplier* performs an acquisition from the *Nature* (milking) and creates a new lot (in the "acquired" state). Then the *Supplier* performs two transformations (souring and ripening): each transformation produces a new lot (in the "transformed" state). Finally, the *Supplier* provides (transport) the *Shop* with the cheese and generates a new lot (in the "provided" state). The *Shop* performs an acquisition, which produces a new lot (in the "acquired" state). When the *Shop* provides (sale) the cheese to the *Consumer*, it creates a new lot (in the "provided" state). The *Consumer* is the last actor of the supply chain: he/she does not create any lot because his/her acquisition has not to be traced.

4. Traceability and e-business standards

In order to implement the data model previously described, we must consider that each responsible actor actually belongs to a company involved in the supply chain. The flow of product lots through the supply chain is associated with information exchanges among responsible actors and possibly third-party organizations. An abstraction of traceability information systems can envision a massive, centralized database capturing in a single location all the information about each lot at each stage of the supply chain. The actual implementation of a centralized solution typically relies on the so-called *push* model [9]: as soon as each responsible actor collects all data relevant to traceability, it pushes these data into the centralized database. Though the push model is very simple, it is not certainly efficient. First of all, it requires that all information about a lot is always available on the centralized database, while data not relevant to traceability could be stored more advantageously at the site where they are generated. Second, it imposes that the centralized database implements all protocols and data formats used in the different stages of the supply chain. The push model appears particularly inadequate in the food supply chain, which is characterized by a number of peculiarities such as [10]:

i) Inhomogeneous structure and naming of data. For several years, important agricultural communities have wrestled with the task of identifying the relevant types of data that should be captured and stored in an agricultural database for a given commodity, and generating a standard naming convention for each data element in that database. Producers have been unsuccessful in building consensus for any single standard for any single commodity, and there is no reason to believe that consensus will ever be reached.

ii) Confidentiality and control of data. Food chain participants, at all segments of production, are often highly protective of their own data, thus they would not agree on sharing their company's data on the same server with the others. Thus, a centralized database would create issues of data confidentiality and trade disruption. Ownership, movement and location data might be used for purpose other than the goal of traceability. Further, there are potential data integrity issues.

The architectural solution which is achieving widespread consensus is to distribute the traceability information among different robust databases along the supply chain, and create a backbone of connectivity between these databases. Actually, the system would not need to operate with constant connectivity. Data may be held locally either within the management system of each actor of the supply chain or associated with the lot itself. Thus, different actors can use different structure and naming of data and agree on a common vocabulary only when interaction is required. Further, each actor is responsible for confidentiality of its data and will provide the other actors with only the information concerning the traceability. Typically, the distributed architecture uses intermediate data trustees. A data trustee is a private, third party intermediary to which an actor transfers its location and ownership data: the data trustee manages these data in place of the responsible actor and handles traceability issues in the case of possible health investigation.

In a distributed architecture, actors have to manage multiple interconnections and deal with multiple interfaces. For instance, a fast food outlet has to cope with the meat, baked goods, dairy products, lettuce, tomatoes and catsup suppliers. It is obvious that it would be preferable to access a single system which provides all necessary information.

The best solution is to build independent, private data sharing networks that are very loosely interconnected [10]. Typically, these networks focus on a certain class of food products. Ideally, there should be as many networks as product classes. Each network would operate autonomously and would be loosely linked with other networks via technology that makes the system appear a single one to a downstream actor without exposing the data from one independent system to another.

The implementation of the private data sharing networks relies on standard inter-organizations co-operation models and protocols. Such models and protocols have been already studied in the framework of electronic business over the web. In particular, the proliferation of XML-based business interchanges has served as the catalyst for defining a new global paradigm that could allow all business activities to be performed electronically. Such paradigm, denoted electronic business eXtensible Markup Language (ebXML) [7], is an international initiative established by the United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT) [11] and the Organization for the Advancement of Structured Information Standards (OASIS) [12].

ebXML represents a set of modular business collaboration-oriented specifications. Business collaboration encompasses the concept of mutually accepted trading partner agreements as well as the concept of a technical infrastructure which enables businesses to locate each other and provides for reliable and secure exchange of business messages between collaborating business partners. ebXML requires collaborating partners to mutually agree upon the formats and semantics of business documents, which are XML-encoded. In an inter-enterprise business collaboration scenario, both business partners should use the ebXML Message Service (ebMS) to securely and reliably transport business documents. ebMS is defined as a set of layered extensions to Simple Object Access Protocol (SOAP) and SOAP Messages with Attachments (SOAPAttach) specifications [13], which are defined by the W3C organization [14].

However, ebMS just represents the message envelope and requires an additional standard to define the semantics of a business document, which constitutes the content of the envelope. Since there are several horizontal and vertical content standards in existence, a novel initiative, called Universal Business Language, is achieving a universal XML business language over ebXML. Of course, ebXML-based business collaboration has to take failure conditions into account too. Transport level failures are managed by ebMS, which caters for reliable and recoverable message exchange. The Business Process Specification Schema (BPSS) handles the business level failures. For example, if a party fails to reply within a pre-defined time period, then the BPSS reverts to the previous known secure state. The message-exchange agreement between two business partners is described by a Collaboration Protocol Agreement (CPA). The CPA can be regarded as the description of the interface of a business service. If a business partner changes the interface described in the CPA, it invalidates the CPA and requires a new CPA to be built. The technical message exchange is not however affected and the sender is still ensured that the message is delivered, delegating the recipient to deal with the potential problem. ebXML has its major strengths when it comes to inter-enterprise business process integration. However, ebXML is also suitable for intra-enterprise business process integration in that functional units (e.g. divisions) are treated as separate mini-enterprises. In B2B scenarios, ebXML is used for managing enterprise-spanning business transaction services in the context of collaborative business [15].

The ebXML technology has been used as reference specification to define and exchange business documents in the interoperability architecture developed for food traceability in the framework of the *Cerere* project. In particular, *Cerere* adopts the Hermes Message Service Handler (MSH) implemented by the Center for E-Commerce Infrastructure Development at the University of Hong Kong [16]. Hermes MSH is in compliance with ebMS standard and includes message packaging, reliable messaging, message ordering, error handling, security, synchronous reply, message status service, and RDBMS persistent storage. In addition to secure and reliable messaging functions, Hermes MSH supports the concept of "quality of service" by respecting the CPAs defined between collaborating actors.

To define the XML documents exchanged by using the Hermes MSH, we adopted the following generally agreed rules: UML objects and attributes are translated into XML elements if they are structured or can assume a large set of values; on the other hand, UML objects and attributes are translated into XML attributes if they are not structured, for instance string or number, and can only assume a set of predefined values.

The set of *Cerere* messages is composed of five XML document types concerning the communication of the state transitions (*acquisition*, *providing* and *transformation*) and the specification of new instances of Lot and Activity. Figure 10 shows the essential structure of an XML document instance of acquisition, providing and transformation notification in the hypothesis of a distributed architecture. Since a distributed architecture requires a minimum exchange of information between the actors and the data trustee, the ebXML messages just carry the identifiers of lots and responsible actors. Figure 11 shows the fundamental elements of an XML document instance of an Activity and a Lot. For the sake of readability, the XML structures of quality feature are shown in Figure 12, for numerical and categorical types, respectively.

5. Conclusions

We have presented a generic data model for food traceability. We have described the basic classes of the model and the patterns used to represent the dynamic behavior of product lots along the food supply chain. We have developed a prototype of a traceability system which is able both to trace back and to trace forward product units and batches. The system also supports quality information. We are currently experiencing the application of the prototype to a real vegetable supply chain in Tuscany, Italy.

```
<acquisitionNotification>
  <fromActorId>
    A001
  </fromActorId>
  <fromLotId>
    L033
  </fromLotId>
  <toActorId>
    A009
  </toActorId>
  <toLotId>
    L033
  </toLotId>
  <date>
    2004-04-15 16:20:19
  </date>
</acquisitionNotification>
```

a)

```
<providingNotification>
  <fromActorId>
    A001
  </fromActorId>
  <fromLotId>
    L033
  </fromLotId>
  <toActorId>
    A009
  </toActorId>
  <date>
    2004-04-15 16:20:19
  </date>
</providingNotification>
```

b)

```
<transformationNotification>
  <actorId>
    A001
  </actorId>
  <fromLotId>
    L033
  </fromLotId>
  <toLotId>
    T047
  </toLotId>
  <date>
    2004-04-15 16:20:19
  </date>
</transformationNotification>
```

c)

Figure 10. A simplified XML document instance of acquisition (a), providing (b) and transformation (c) notification.

```
<activity type ="purchase">
  <id>A055</id>
  <respActorId>A009</respActorId>
  <startingDate>
     2004-04-15 16:20:19
  </startingDate>
  <duration unit ="hour">1</duration>
  <siteId>S007</siteId>
  <qualityFeature>...</qualityFeature>
  <generatedLot>
     <id>T047</id>
  </generatedLot>
  <componentLots>
     <id>L033</id>
     <respActorId>A009</respActorId>
  </componentLots>
</activity>
```

a)

```
<lot type ="Wine Cask">
   <id>T047</id>
   <respActorId>A009</respActorId>
   <generationDate>
      2004-04-15 16:20:19
   </generationDate>
   <siteId>T038</siteId>
   <activityId>A005</activityId>
   <qualityFeature>...</qualityFeature>
</lot>
```

b)

Figure 11. A simplified XML document instance of an Activity (a) and a Lot (b).

```
<qualityFeature>
   <description>
     color intensity
   </description>
   <numericalQF
      unitName ="Intensity"
      minValue ="1"
      maxValue ="10">
      <value>8.21</value>
   </numericalQF>
</qualityFeature>
```

a)

```
<qualityFeature>
   <description>ratings</description>
   <categoricalQF>
      <value>2 stars</value>
      <categoricalValue
         value ="1 stars"
         ordering ="0"
         description ="good"/>
      <categoricalValue
         value ="2 stars"
         ordering ="1"
         description ="very good"/>
      <categoricalValue value ="3 stars"
          ordering ="2"
          description ="excellent"/>
   </categoricalQF>
</qualityFeature>
```

b)

Figure 12. An XML translation for numerical (a) and categorical (b) quality features.

Acknowledgments

The Cerere project is partially supported by The Foundation Cassa di Risparmio di Pisa under contract 2003.0132

References

[1] Regulation (EC) n. 178/2002 of the European Parliament and of the Council of 28 January 2002, ch. V.

[2] U.S. Food and Drug Administration, regulation 21CFR820, "Title 21: Food and drugs, subchapter H: Medical devices, part 820 Quality system regulation", revised April 1, 2004, http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfcfr/CFRSearch.cfm?CFRPart=820.

[3] Ministry of Agriculture, Forestry and Fisheries of Japan, “Guidelines for Introduction of Food Traceability Systems”, March 2003.

[4] Food Standards Agency, “Traceability in the Food Chain – A preliminary study,” March 2002, http://www.foodstandards.gov.uk/news/newsarchive/traceability.

[5] M. de Castro Neto, M.B. Lima Rodrigues, P. Aguiar Pinto, I. Berger, “Traceability on the web – a prototype for the Portuguese beef sector”, in Proc. of EFITA 2003 Conference, Debrecen, Hungary, 5-9 July 2003, pp. 607-611.

[6] C.A. van Dorp, “Tracking and Tracing Business Cases: Incidents, Accidents and Opportunities”, in Proc. of EFITA 2003, Debrecen, Hungary, 5-9 July 2003, pp. 601-606.

[7] ebXML official website – http://www.ebxml.org.

[8] H.M. Kim, M.S. Fox, M. Gruninger, “Ontology of Quality for Enterprise Modelling” in Proc. of WET-ICE, Los Alamitos, CA, USA, 1995, pp. 105-116.

[9] W.R. Pape, B. Jorgenson, R.D. Boyle, J. Pauwels, “Let’s get animal traceback right the first time”, Food Traceability Report, Feb 2004, pp. 14-15.

[10] W.R. Pape, B. Jorgenson, R.D. Boyle, J. Pauwels, “Selecting the most appropriate database architecture”, Food Traceability Report, Feb 2003, pp. 21-23.

[11] UN/CEFACT – United Nations Centre for Trade Facilitation and Electronic Business, http://www.unece.org/cefact/.

[12] OASIS - Organization for the Advancement of Structured Information Standards, http://www.oasis-open.org.

[13] SOAP, http://www.w3.org/TR/soap/.

[14] W3C – World Wide Web Consortium, http://www.w3.org/.

[15] D.E. Jenz, “ebXML and Web Services - Friends or Foes?”, 27.06.2002, http://www.mywebservices.org/index.php/article/articleview/451/1/1/.

[16] Hermes Message Service Handler, http://www.freebxml.org/msh.htm.