Patterns and technologies for enabling supply chain traceability through collaborative e-business

Alessio Bechini, Mario G.C.A. Cimino, Francesco Marcelloni *, Andrea Tomasi

Dipartimento di Ingegneria dell'Informazione: Elettronica, Informatica, Telecomunicazioni, University of Pisa, Via Diotisalvi 2, 56122 Pisa, Italy

Abstract

Industrial traceability systems are designed to operate over complex supply chains, with a large and dynamic group of participants. These systems need to agree on processing and marketing of goods, information management, responsibility, and identification. In addition, they should guarantee context independence, scalability, and interoperability. In this paper, we first discuss the main issues emerging at different abstraction levels in developing traceability systems. Second, we introduce a data model for traceability and a set of suitable patterns to encode generic traceability semantics. Then, we discuss suitable technological standards to define, register, and enable business collaborations. Finally, we show a practical implementation of a traceability system through a real world experience on food supply chains.

Keywords: Traceability; Inter-organizational systems; e-business; Information systems; Service oriented architecture; ebXML; Web Services

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 supply chain. In practice, chain traceability is achieved if businesses keep records of suppliers and customers and exchange this information along the entire supply chain. In particular, each unit/batch (called *lot* in the following) of a component 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 [1]. Traceability is a needed strategic service in any production context. It can be used to improve security, control quality, combat fraud or manage complex chains [2]. In particular, traceability in food supply chain has attracted considerable attention in the last few years for a variety of reasons [3]. First of all, it has become a legal obligation within the EU since 1st January 2005 [4]; similar requirements for traceability systems are present in the United States [5] and Japan too [6]. 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 company image and in the specific product. Consequently, 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 also take food safety and quality improvement into account [7]. This means, for example, enabling the system to trace back so as to discover the cause of a problem and to prevent it from happening again, or to trigger a proper recall of potentially unsafe products, thus protecting public health. Of course, the implementation of a complete and efficient traceability system has to cope with several problems, such as the lack of alignment of the possibly different systems adopted in

* Corresponding author. Tel.: +39 050 2217 678; fax: +39 050 2217 600.
E-mail addresses: a.bechini@iet.unipi.it (A. Bechini), m.cimino@iet.unipi.it (M.G.C.A. Cimino), f.marcelloni@iet.unipi.it (F. Marcelloni), a.tomasi@iet.unipi.it (A. Tomasi).

the various segments of the supply chain, or the non-homogeneous information kept at the various supply chain units [8,9].

Building 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 movement, transformation or combination) that operate on products. To this aim, traceability needs to be supported by appropriate architectural and technical implementation solutions, as well as suitable operational services, in order to provide its expected value for business partners. These solutions have been studied in the framework of virtual organization (VO) models [10,11], where independent organizations share resources and skills to achieve a specific goal. In this context, Choudhury [12] has analyzed problems from the standpoint of a firm making strategic decisions about inter-organizational systems (IOSs), addressing the questions of what types of IOSs might be useful, and how these IOSs might be developed. By extending the typology based on the transaction cost economics proposed by Malone et al. [13], Choudhury describes three types of IOSs architectures: electronic monopolies, multilateral IOSs, and electronic dyads. The increasing feasibility of adopting a peer-to-peer (P2P) approach for business-to-business (B2B) collaborations decreases the need for centralized exchanges, making electronic dyads more and more attractive. B2B based on P2P allows implementing dynamic electronic dyads from the IOS perspective [12]. Indeed, Silva et al. [14] pinpoint that, in any implementation of a VO model, dynamic reconfigurability, and business alignment with the market requirements can be considered as the most important interrelated aspects. After a broad review of the offerings of key e-marketplace makers, they observe that the compliance towards Electronic Data Interchange (EDI) [15] is often guaranteed; despite of this, they also recognize Web Services (WS) [16] and electronic business using eXtensible Markup Language (ebXML) [17] as the most promising technologies for the creation of dynamic collaborative environments and business process integration. In Gunasekaran et al. [18], VO models are based on the outsourcing concept to take advantage of the core competencies with the objective of being flexible and responsive to changing market requirements. Thus, companies integrate various links of the supply chain and their supporting information systems: such integration is driven by the need to streamline operations. Types of architectures, dynamic reconfigurability, business alignment, dynamic collaboration, business process integration, flexibility, and responsiveness are therefore some of the main aspects that have to be considered when developing IOSs.

This paper proposes an organic approach to manage the aforementioned aspects inherent in inter-organizational information systems and relevant technical aspects specific to traceability (such as traceability semantics, scalability, information management, and lot identification) in the development of a traceability system. These aspects have been widely discussed in the literature, but often they have been tackled separately, proposing generic patterns independent of the specific application domain. In this context, our approach aims to enable existing models and technologies, and to create new domain-specific patterns in order to develop an effective traceability system. The paper is structured as follows. In Section 2, we introduce a data model for describing assets and actors. Then, in Section 3, we show a formal description of the lot behavior throughout the supply chain. In data exchange, a crucial role is played by lot identification. Section 4 introduces some of the most important techniques, such as barcodes and Radio Frequency Identification (RFID), and standards, such as GS1 and Electronic Product Code (EPC), for automatic lot identification, focusing on their potential contributions in reducing the cost of procedures for tracking goods. The structure of a traceability system depends on how data are managed by the involved actors: the possible choices are described in Section 5. Independently of the type of structure, traceability relies on an integrated environment. Section 6 discusses how this integration can be achieved by exploiting recently proposed middleware solutions, like Enterprise Service Bus (ESB). Furthermore, as the system reliability heavily depends on agreed business interfaces among the supply chain partners, Section 7 is devoted to the discussion of business process interoperability, through enabling technologies, like WS, and standards for inter-enterprise business collaboration, like ebXML. In this context, the Service Oriented Architecture (SOA) model is presented as a key paradigm. Finally, we describe a real world experience with food and beverage companies in Section 8, where the most important XML artifacts and the system architecture are presented.

2. Static structure of a traceability system

As previously stated, a traceability system must be able to trace both lots and activities [19]. This means that each data model related to traceability must include lot and activity as key entities, and allow lot tracking and tracing [1]. *Tracking* refers to the ability to follow the downstream path of a product along the supply chain, possibly according to some specific criteria. This is a crucial factor, e.g., for an efficient recall of faulty 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. Fig. 1 shows a typical scenario of a product recall due, e.g., to a contamination event in a simplified supply chain consisting of only four segments. In the figure, a box denotes a *traceability lot* (*lot*, for short), that is a product unit 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 gear represents an *activity*, such as production, packaging, distribution, and sale, which may receive N lots as input and

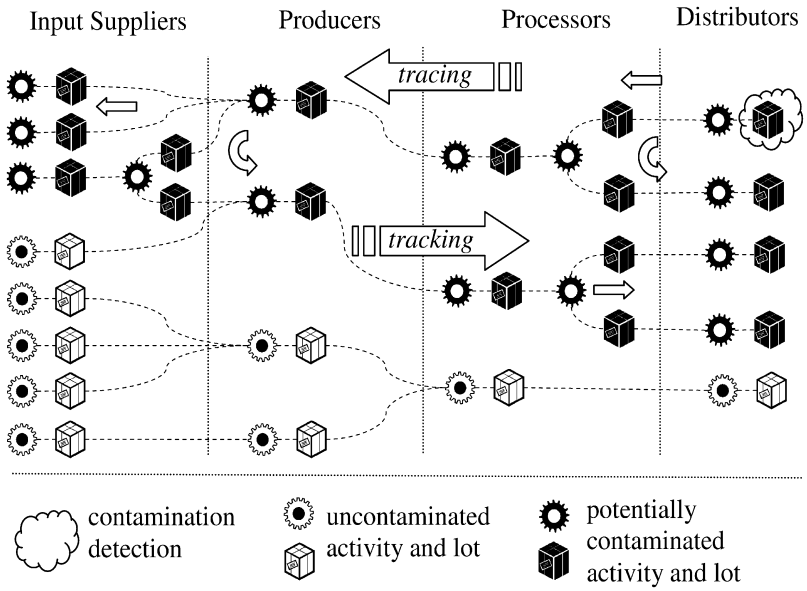

Fig. 1. Typical scenario for a product recall in a supply chain.

may deliver M lots as output. A dashed 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 Fig. 1, supply chain units are input suppliers, producers, processors, and distributors. 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.

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 Fig. 1 requires the adoption of an appropriate *data model*, which must be general enough to represent any kind of product. It also has to provide a means to univocally identify traceability lots and activities, and to record information about lots and activities, and their relations. Further, it is often recommendable to take explicitly into account additional data on quality features. For example, in a cooking activity, the oven temperature and humidity could be important parameters to monitor potential hazard situations. To formally describe the different aspects of the modeled system, in this paper we adopt the standard Unified Modeling Language (UML) notation [20]. UML is a general-purpose visual modeling language that can be proficiently used to specify, visualize, construct, and document the artifacts of an information system. UML supports a number of diagram types, i.e., graphical presentations of model elements, most often rendered as a connected graph.

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 GS1 [21] standard. Each actor is assumed to be uniquely identified in the supply chain by an *actor identifier*. Moreover, an actor is allowed to freely associate an identifier (*traceable entity identifier*) with each traceable entity (i.e., either an activity or a lot) the actor is responsible for. If an actor creates several distinct products, the lot identifier may consist, for instance, of the product type identifier and one 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*.

Fig. 2 shows the data model. Here, classes are grouped into two distinct UML packages: *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 (i.e., pertaining to the so-called *product information*). The *TraceableEntity* is an abstract class that models the basic characteristics of the two entity types involved in traceability: lots and activities. The field *TraceableEntity.id* implements the traceable entity identifier. The association *"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, *TraceableEntity* is also associated with *Site*, which holds its own unique identifier: i.e., each lot is placed in one 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 *ResponsibleActor* are characterized by a number of attributes that summarize all the information required for traceability. The

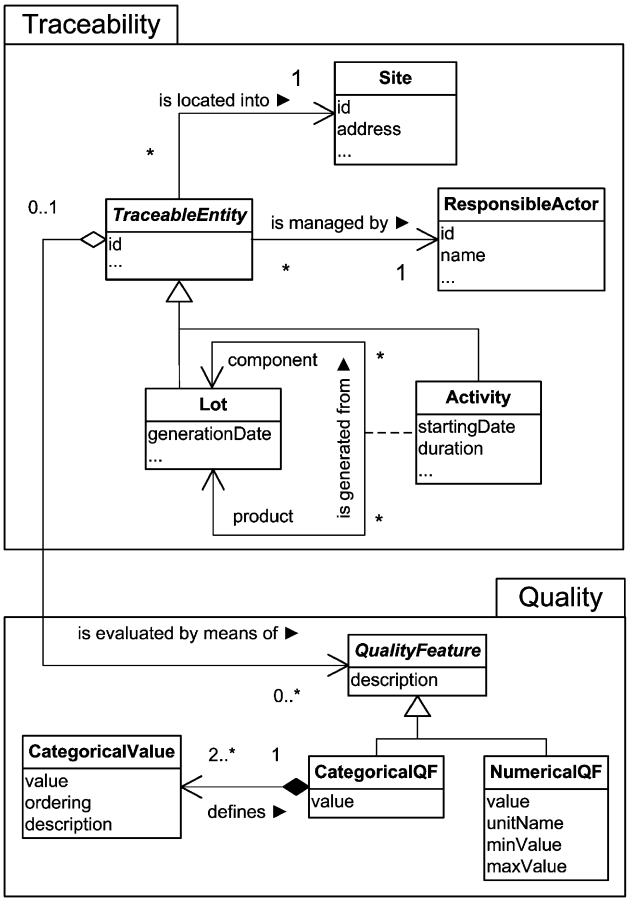

Fig. 2. UML class diagram of the traceability data model.

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

Fig. 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. Thus, several different instances of class *Lot* can correspond to a unique physical lot.

In Fig. 4, two UML sequence diagrams describe two possible message exchanges within a purchase action. In the first diagram, we refer to a distributed model with no central tracking management. Here, the actor 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, who is in charge of binding such an identifier to the other corresponding identifier for the output lot. This association allows both lot tracing and lot tracking. Typically, the global identifier is attached as barcode or RFID tag to the lot. Thus, part of the communication consists of reading lot identifiers (by means of appropriate appliances) at successive supply chain units.

In the second diagram, there exists a central tracking manager, which is responsible for the traceability data. This model requires that each supply chain actor responsible for an activity provides the tracking manager with all the information related to the activity. In particular, this information must allow the data management system at least to associate the input lot(s) with the output lot(s). In both the models, in order to retrieve the history of a lot, each actor of the supply chain has to communicate with other actors: with its trading partners in the first model and with the tracking manager in the second. In fact, legally, the requirement [4–6] for traceability is limited to ensure that businesses are at least able to identify the immediate supplier of the lot and the immediate subsequent recipient ("one step back-one step forward" principle), with the exemption of retailers to final consumers. The data exchange must of course be carried out in a secure and reliable way.

Quality requirements often play a crucial role in modern business process management, and thus they deserve particular attention in the corresponding traceability systems as well [7]. The ISO 9000 standard [22] 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 Fig. 2. This package contains the abstract class *QualityFeature* (QF), which includes a description of the feature itself and a collection of methods to set and retrieve feature values. Values can be either cat-

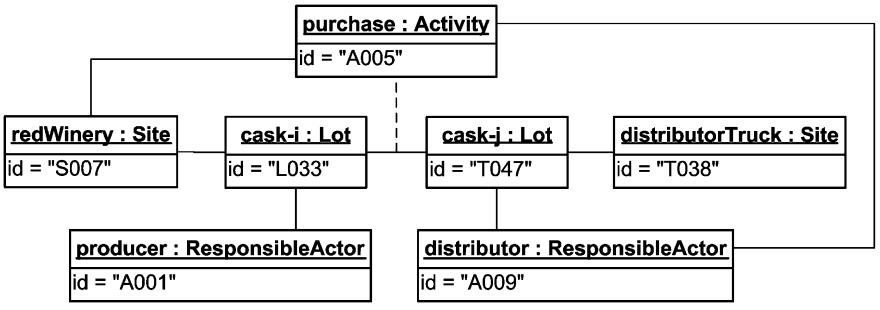

Fig. 3. Objects involved in recording the actual execution of a simple activity.

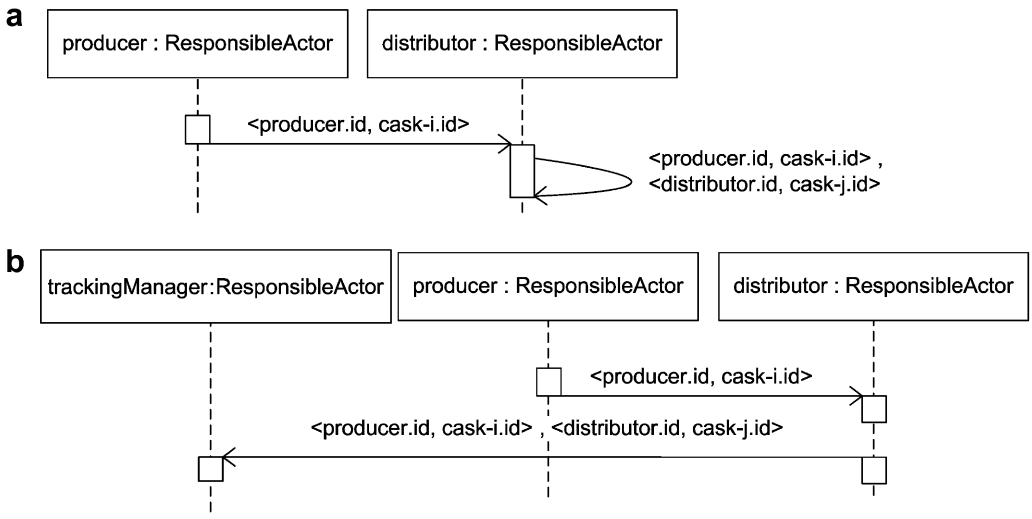

Fig. 4. Sequence diagram of a purchase activity; (a) distributed model (b) centralized model.

egorical or numerical. *CategoricalQF* and *NumericalQF* concrete classes implement features that can assume, respectively, categorical and numerical values. *CategoricalQF* contains a set of *CategoricalValue* objects, which define the possible values. A *CategoricalValue* is characterized by the value, a description, and an ordering value. This last item can be used whenever ordered categorical values are needed. *NumericalQF* 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.

Fig. 5 shows an example of object diagram that describes the quality features "color intensity" and "rating" associated with lot *cask-i* of wine bottles. 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".

The full comprehension and monitoring of what actually happens along the supply chain requires not only a precise data model for the involved assets, but also a clear understanding of the lot temporal progression towards successive stages in the supply chain. In a nutshell, a simple formal characterization of the lot "history" is needed for the investigation on the actual requirements of the overall traceability system. The key observation is that the lot progression is determined by *activities over it*, and thus its behavior can be described according to a classification of the activities that a generic lot may undergo. These aspects are discussed in next section.

3. Basic behavioral patterns in a traceability system

The lot behavior can be modeled by the following six activity patterns [6], as shown in Fig. 6a–g using a UML activity diagram. For the sake of simplicity, we have omitted the object *Activity* as output of the UML activity. This object maintains the relation between the pre-activity lot and the post-activity lot.

1. *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. Real examples of lot integration are mixing and packing. Fig. 6a shows a scenario of integration with three pre-integration lots concurrently integrated into a unique post-integration lot.
2. *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. Real examples of lot division are cutting and splitting. Fig. 6b shows a scenario of the lot division pattern.
3. *Lot alteration*: As shown in Fig. 6c, 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. Real examples of lot alteration are heating, freezing, and drying.
4. *Lot movement*: A lot is moved from one storage site (source site in Fig. 6d) to another (destination site) 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.
5. *Lot acquisition*: An actor (*buyer* in Fig. 6e) of the supply chain acquires a lot from another actor (*provider*). Since a lot can have only one responsible actor, the buyer

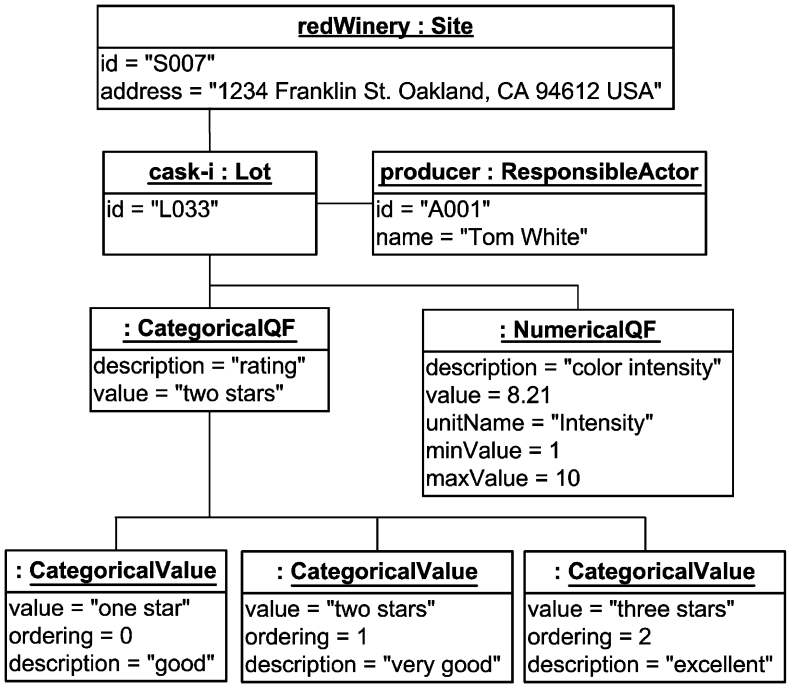

Fig. 5. Example of objects related to quality features.

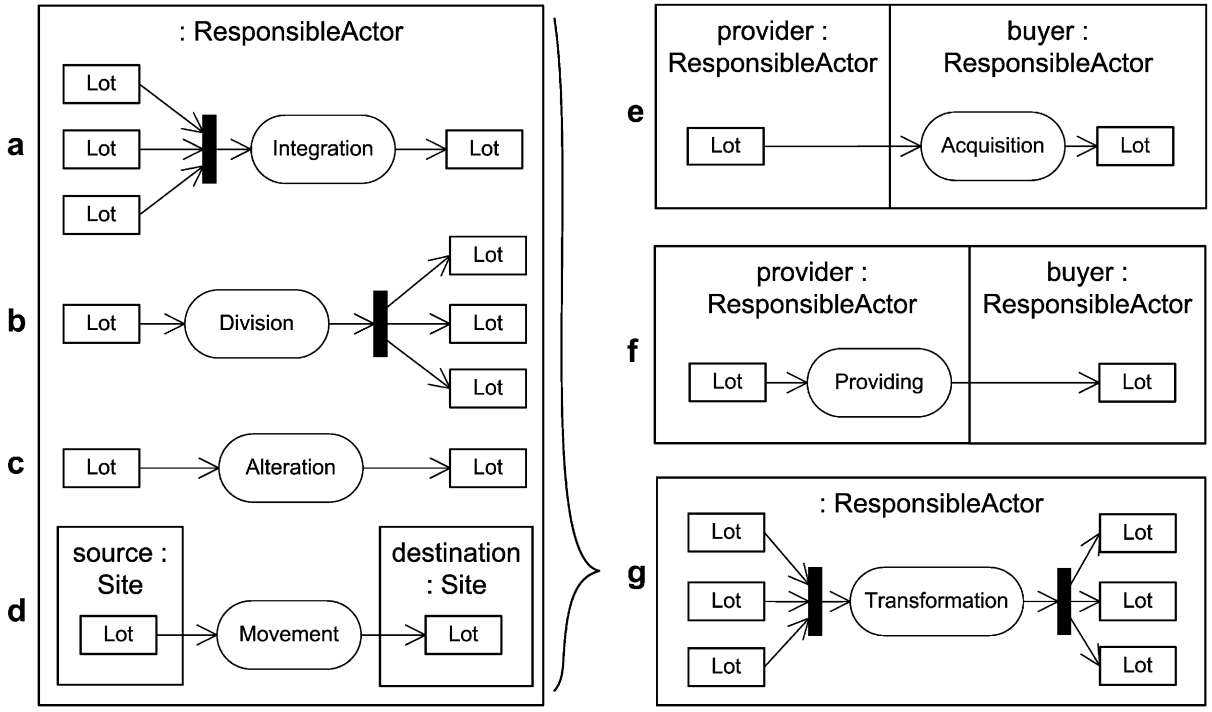

Fig. 6. Basic behavioral patterns for a lot.

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.

6. *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. Fig. 6f shows the lot-providing pattern.

From a dynamic point of view, the lot integration, lot division, lot alteration and lot movement can be modeled as a generic lot *transformation* from N lots to M lots (see Fig. 6g for a scenario of transformation with three input and output lots). Thus, a lot division into N separate 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 *transformation* pattern, in the *acquisition* and *providing* patterns the incoming and outgoing lots have distinct associated responsible actors. Transformation of a lot can occur only if the responsible actor of the transformation is also responsible for the lot. This implies that the responsible actor has to acquire a lot before undergoing any transformation. But the existence of an acquiring actor implies the presence of a providing actor for the lot as well. Further, a lot that has been acquired (provided) cannot be acquired (provided) again before being provided (acquired). A description of this behavior is shown in Fig. 7, by means of a UML activity diagram.

In the figure, we introduced two particular responsible actors, namely *nature* and *customer*, 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. The *Acquisition* activity is triggered by an extraction from nature or when lots are provided by a *Providing* activity. Only lots, which already belong to the responsible actor, can be transformed or provided. Lots produced by a *Transformation* activity can be transformed in their turns or trigger a *Providing* activity. Lots generated by a *Providing* activity can be either discarded, or consumed or provided to another responsible actor.

As an example, let us consider a simplified cheese supply chain. The starting point of the supply chain is the milk acquisition (from nature, in our simplified setting). In the first place, milk is soured with lactic acid bacteria by the supplier. After milk thickening, the produced gelatin is reduced to small pieces with cutting and mixing tools, and then it separates into curds (the solid components of the milk) and whey (the water contained in the milk). At this stage, the producer must decide to make soft, cut or hard cheese. The cheese type is determined by the curd size, temperature, pressure, etc. The cheese curd is put into forms and pressed, according to the desired cheese type. Then, the salting process is carried out in order to support the rind development. Finally, the cheese is placed into special ripening rooms: the ripening process is controlled by the humidity in the air, temperature, and maintenance of the cheese surface.

This simple supply chain can be modeled as depicted in the UML communication diagram shown in Fig. 8. Here *nature*, *supplier*, *shop*, and *customer* are the different *ResponsibleActors*, and they interact according to given activities, possibly producing new lots. The activity ordering is specified by the numbers associated with the shown procedures.

At the beginning, the *supplier* performs an acquisition from the *nature* (milking) and creates a new lot. Then the *supplier* performs two transformations (souring and ripening): each transformation produces a new lot. Finally, the *supplier* provides (transport) the *shop* with the cheese and generates a new lot. The *shop* performs an acquisition (buying), which produces a new lot. When the *shop* provides (sale) the cheese to the *customer*, it creates a new lot. The *customer* comes after the last responsible actor of the supply chain: he/she does not create any lot because his/her acquisition has not to be traced.

As highlighted in the last example, the tracking process along the chain production possibly generates a huge amount of data records. In order to allow tracing procedures to remotely retrieve such data, a proper identification technology is needed. This aspect is detailed in next section.

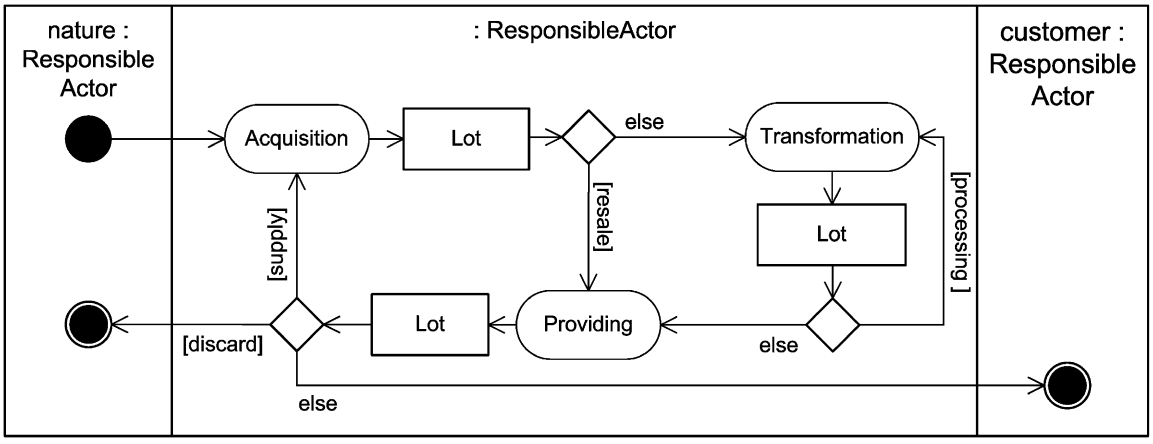

Fig. 7. UML activity diagram describing the lot behavior.

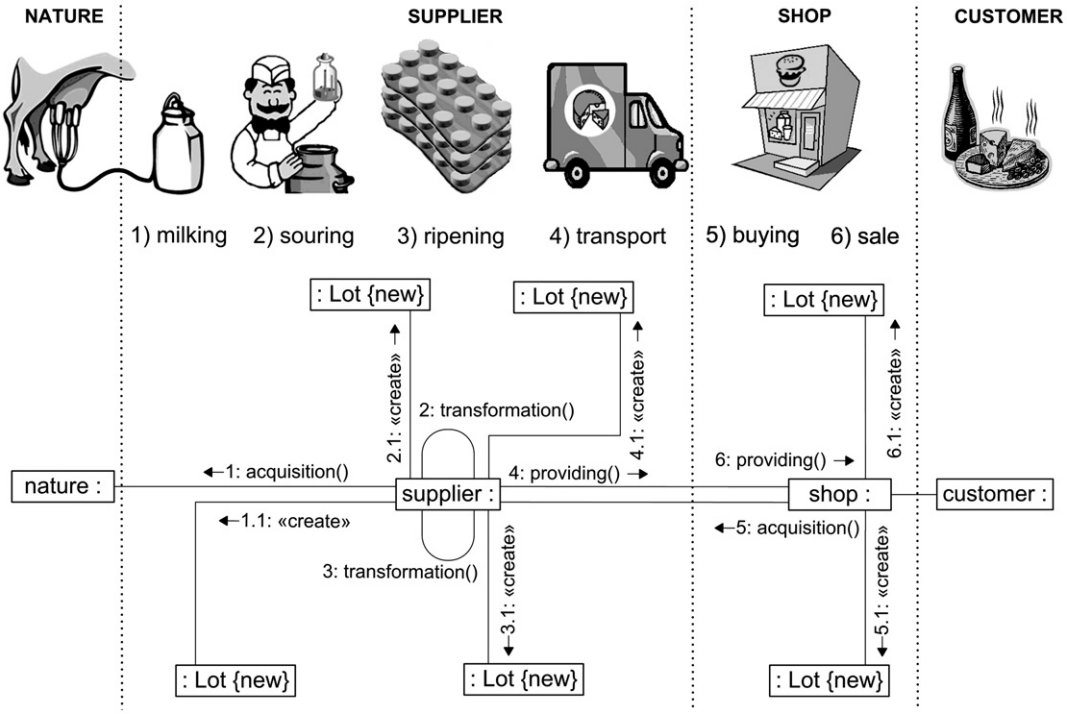

Fig. 8. Communication diagram for a simplified traceability system in cheese production.

4. Lot identification

Several experiences in the context of food traceability have shown that the ability to generate accurate information typically improves at the later stages of production and processing. On the other hand, the closer to dirt one gets, the more difficult collecting accurate data is. As regards the simplified cheese production shown in Fig. 8, for instance, the information is certainly more accurate at the last stages (in the communication from supplier to shop) than at the first stage (acquisition of the milk from nature). This problem, known as the "first mile" problem, can considerably reduce the benefits of a traceability system. Since the easiest place to capture data is when the event or transaction takes place on the farm or in the field, capturing data first hand is better than reconstructing it afterward. Time and distance from the point of origination only serve to reduce the validity of data and increase the total lifecycle cost of managing it [23]. Further, whenever possible, data should be collected without human intervention. Humans are typically poor data collectors. Several techniques have been proposed to collect data in a cost-effective manner without interrupting the workflow. A widely accepted technique is to associate a tag with each lot: the tag contains the data necessary to identify the lot. These data can be automatically read by appropriate readers.

The most used tags are barcodes and RFID tags. RFID tags consist of a chip that can be attached onto or implanted into any surface of an item [24]. As regards their employment for traceability issues, RFID technology looks very promising [25]. Unlike barcodes technology, e.g., RFID allows acquiring information at a rate of 1000 tags per second. Thus, it is reasonable to expect a growing acceptance of RFID technologies in the next years as basic components within traceability information systems.

The simplest form of identification consists in a numeric or alphanumeric string. The string gives no information about the unit, but provides a univocal key to retrieve traceability data stored elsewhere. To guarantee the string uniqueness, several standard systems have been introduced. The most promising one is certainly the GS1 (formerly EAN.UCC) system [2,21]. By administering the assignment of company prefixes and coordinating the accompanying standards, GS1 maintains the most robust lot identification system in the world. GS1 provides seven Identification Keys to support the identification of items, services, locations, logistic units, returnable containers, etc. As regards traceability, most of the numbering structures of interest have been developed by GS1; among them, GTIN (Global Trade Item Number), which identifies uniquely each commercial unit, SSCC (Serial Shipping Container Code), which identifies uniquely a logistic unit (dispatch unit), GLN (Global Location Number), which identifies any legal, functional or physical location within a business or organizational entity [2]. More complex forms of identification can however be realized, by introducing descriptions of the key features of the item [6]. GS1 provides training and support for another important emerging standard, the EPC technology currently under development at EPCGlobal [26] to sustain the use of RFID tags. The EPC identifier is a meta-coding scheme designed to address the needs

of different kinds of industries. The EPC identifier accommodates existing coding schema (whenever possible) and new schema (whenever it is necessary). Though GS1 and EPC standards are very good candidates for the definition of the global identifier, we decided to allow other alternative methods too, so as to make our system as general and flexible as possible (small enterprises might be reluctant to join the GS1 community, due to the initial difficulties of justifying the investment).

Lot identification is the first step towards an effective traceability information management. The identifier is the key to retrieve data, but how are data records organized? Where are such records stored? How are they communicated among the different actors? In the next two sections, we describe some possible architectural solutions to make traceability effective.

5. Traceability information management within the supply chain

As stated, e.g., by the European Community food law guidance [27], business operators are expected to ensure relevant information requirements and to verify their fulfillment, within the businesses under their control. From an information management perspective, implementing a traceability system within a supply chain requires all the involved parties to systematically bind the physical flow of materials/products to the corresponding information flow. This is typically attained by developing a generic supply chain technical disciplinary, which defines a set of rules with regards to material and document flow, production process management and execution, business process collaboration through partners agreement, and responsibility placements. In the disciplinary, traceability requirements must be stated, as well as quality and safety goals. From an abstract viewpoint, a traceability information system can be thought as a single massive, centralized data storage capturing all the information about each lot along each stage of the supply chain. The logical view of a lot contains attributes related to each feature of every product and its components, as well as details of the processing phases. Any traceability system adopting an actually centralized solution is structured according to the so-called *push* model [28]. This paradigm states that, as soon as each actor in the supply chain collects traceability data, such data must be transferred to the centralized traceability data storage. However, the implementation of one single, centralized traceability data storage is neither realistic nor efficient in most actual settings, characterized by growing dynamism, heterogeneity, complexity, and scale.

In order to foster scalability, in the first place we should keep logically separate traceability information and data from other product/lot characteristics: in fact, although the former may be even managed through a single common system, the latter can be sensibly stored at the different sites where the corresponding measurements take place. Bringing the discussion one step forward, multiple traceability data repositories could be proficiently linked to either different stages or external data trustees, thus obtaining a physically distributed architecture; in this scenario, each different node would possibly address local specific management and implementation issues.

As a reference example for pointing out significant issues in actual management of traceability data, we can consider the peculiarities of a generic food supply chain [29]:

1. Heterogeneous structure and naming of data: For several years, important agricultural communities have wrestled with the task of identifying the relevant information to be captured and stored in an agricultural database for a given product, and developing a standard naming convention for each data element in that database. Producers have failed in building consensus for any single standard for any single commodity, and there is no reason to believe that consensus will ever be reached;
2. 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 own company's data. The industry is concerned that a centralized database would create issues of data confidentiality and trade disruption. Ownership, movement, and location data might be used for purposes different from the goal of traceability. Further, there are potential data integrity issues.

Such concerns further push to find proper alternatives to the employment of a logically centralized solution. The architectural model that is achieving widespread consensus accounts for the distribution of traceability information among different robust data storages along the supply chain, taking also a connectivity backbone between them into account. In such a setting, a constant connectivity is not strictly required to guarantee a proper system operation: in fact, data may be held locally (either within the management system or associated with lots) and transferred just during the periods when communication is possible. Therefore, different actors can use different data structure and naming, agreeing on a common vocabulary *only when interaction is required*. Further, each actor is responsible for confidentiality of its own data, and he will provide the others with traceability information only. Typically, this kind of distributed architecture, based on the so-called *pull model*, also makes use of an *intermediate data trustee* [28]. A data trustee is a private, third party intermediary among responsible actors and towards other entities: companies, government, individuals, or associated consumers. Each actor transfers its location and ownership data to a data trustee. The data trustee acts like an escrow agent, holding the actor's data until a legitimate investigation need arises.

The UML sequence diagram in Fig. 9 introduces the *pull* model: here the data flow is handled in a four-step process.

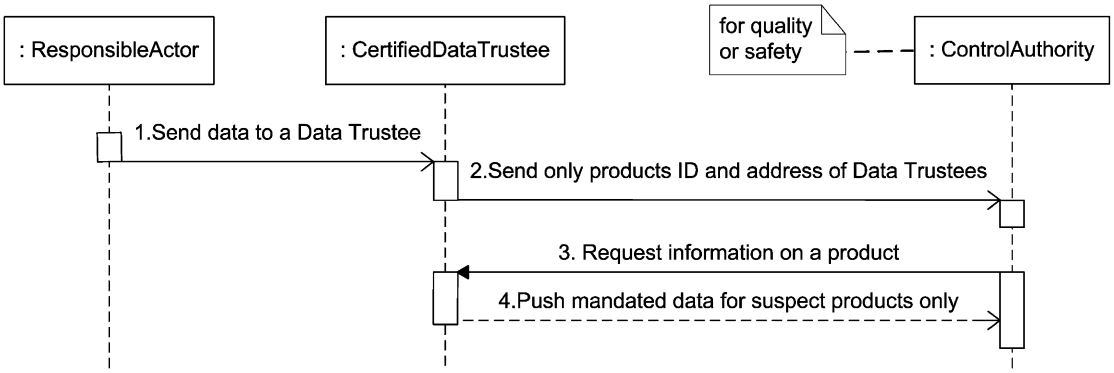

Fig. 9. The pull model: in this paradigm, the responsible actor is expected to send all the required data towards a certified data trustee.

First, the actor inputs data into its private software system normally used to manage the operation. This software system is linked to a data trustee chosen by the actor. Obviously this initial phase can be accomplished manually in case no information system is present. Mandatory data (no commercial or production data) are definitely pushed to the data trustee. Second, the data trustee can expose only the product identification number to external users. No other ownership, location, or movement information is exposed at this point. The third step is possibly taken as authorized users or government officials ask for product information, as in the case of detecting a costumer's health incident. As a consequence of such request, in the fourth step the data trustee publishes to the requester the mandated data, bounded to the suspected product only.

A more realistic scenario would encompass multiple data trustees, being certified and audited by the government or by a government's appointed agency (e.g., a trade association of a certain product class). Actors can freely choose their own data trustees for their own data. Some of the largest actors in the supply chain might even decide to act as data trustees by their own, applying for the related certification: this choice mimics the self-insurance approach many large corporations use for risk management [28].

Furthermore, in the *pull* model, the distributed management approach might be extended to the product identifiers, thanks to the hierarchical structure of the global lot identifier and to the "one step back-one step forward" principle. Indeed, the actor ID, which is encoded in the lot identifier, can be a unique key for a global public registry of responsible actors, which in addition can be hierarchically organized on regional basis. The registry contains the data trustee URI (Uniform Resource Identifier), which actually stores information both on products and on the IDs of ingredients/components, thus safeguarding business information and ensuring factual scalability. Of course, in some contexts it is recommended to carry out system check-ups to determine whether each data trustee is performing its obligations.

The pull architecture has proved to be effective in other critical fields like the global credit card organization. Credit card transactions can occur within a matter of seconds even though the technology must seamlessly link a large number of separate data management systems [30]. This is also the architecture used in the Brazilian national animal identification program, which covers a national herd roughly twice the size of that in the United States [31]. Some producers and processors may still opt to "push" their data in a common repository, e.g., for enhancing the value of their products by publicly showing information about their origin/features, or about the identification with a valuable brand. However, the use of a data trustee will provide an alternative, helping actors protect the confidentiality of their data and the integrity of their existing trading relationships, and increase data integrity within the system.

We can finally state that, within a supply chain, the management of traceability data can be supported by a system built up according either to the push or to the pull models. Although the latter could be regarded as more suitable to address decoupling of the participating actors and protection of data privacy, the final choice between the two models should depend on the size, requirements, heterogeneity, and dynamism of the actual involved supply chain [32].

6. Business process integration infrastructure

Independently of the information management models adopted in the traceability system, actors are asked to cooperate so as to obtain some kind of information shared across different business processes at different locations [8,9]. This cooperation, however, should not be achieved by increasing the effort required to each actor in the supply chain. Each actor has its own approach to manage information and it is not likely enthusiastic on adapting this approach to the other actors. Let us consider, for instance, a retailer that buys products from several different providers. Each single provider is likely to have its information management system different from the other providers' ones. For example, the retailer should not be coerced to interact with a huge number of disparate systems just in order to accommodate traceability requirements. In fact, the retailer must be free to keep its own approach to

information management by exploiting an infrastructure to manage heterogeneous interfaces in a transparent way. Similarly, fast food outlets would not like to use a separate system for their meat, their baked goods, their dairy products, their lettuce, their tomatoes, their ketchup, and so on. Instead, it would be preferable to access a single system able to provide all the required information. In order to achieve this kind of system integration in a heterogeneous setting, a proper infrastructure for business process integration must be set up. A possible solution relies on building up independent, private data-sharing networks (PDSNs), i.e., loosely interconnected data-sharing systems. A PDSN [29] is initiated by one sponsoring company at any production segment and it proceeds linking to individual suppliers and customer companies, thus expanding the initial network. Typically, PDSNs focus on a certain class of products, making issues in data schema definitions simpler. Once built, different independent PDSNs could operate autonomously, yet appearing as a single one to a downstream customer.

In designing a given independent PDSN, specific point-to-point connections should be avoided. In fact, two applications could be easily connected, e.g., using standard application program interfaces (API's), XML [33] data structures and SOAP (Simple Object Access Protocol) [34], as usually done in WS [16], but this simple solution has a number of shortcomings. In particular, scalability issues must be taken into account: in case of n applications, $n(n-1)/2$ (possibly) different interfaces have to be developed. We observe that n is actually larger than the number of production segments, because at each segment there could be more than one pre-existing application program to be involved (e.g., procurement system and separate manufacturing system). The simple adoption of point-to-point connections hampers maintenance as well, because a single change in the application determines the substitution of all the corresponding interfaces.

Recently, the *Enterprise Service Bus* (ESB) [35] has been proposed as an architectural scheme for connecting third-party applications, and such a solution fits also the requirements for our PDSNs. ESB is an integration middleware, standards-based, service-oriented backbone capable of connecting hundreds of application endpoints. ESBs combine messaging, WS, XML, and data transformation/management, in order to reliably connect and coordinate application interactions. In our scenario, ESB would translate data from one third-party system (say a *producer* system) to an internal *data bus* format, and then would retranslate this information to another third-party system. The *data bus* approach requires developing only one interface between each third-party application and the *data bus*, thus reducing the integration complexity. Further, application and business changes can be easily managed within the ESB infrastructure. Schema mediation problems [36], however, may arise in the ESB approach as well, because each company must both publish its data in a common language (e.g., XML) and agree on the naming convention for each data element. Further, if the supply chain spans different countries, the use of different languages makes the problem even worse. These observations suggest transferring most of the naming translation responsibility to the ESB. This last choice definitely simplifies the overall system, allowing each application program to use its own terminology.

Fig. 10 shows a UML component diagram of an ESB for traceability purposes. A *ResponsibleActor*, a *ControlAuthority* or a *DataTrustee* interact with each other using the own interfaces via the ESB.

In Fig. 10, the *DataNamingTranslation* component provides the translation of a data element from one system to another on the basis of wording conventions. The component concerns the mapping of general terminologies implemented via syntactic or lexical rules that are not related to a specific business context, for instance, synonyms and homonyms resolution. The *DataNamingTranslation* allows each system to adopt the own terminology, avoiding possible misunderstandings and reducing the complexity of the information systems. The *Message-Oriented Middleware Facilities* (MOMFacilites) component supports any asynchronous data exchange. The need for this component derives from the lack of standards ruling the use of MOM. All the major vendors have their own implementations, each with its own application programming interface (API) and management tools. One of the most used APIs is JMS (Java Message Service), provided with the Java Enterprise Edition platform and implemented by the majority of MOM vendors [35]. The various components hooked into the ESB have their own expectations of data formats, and these might differ from other components. The *TransformationServices* component makes it possible to ensure that data received by any component is in the format it expects, thereby removing the need to make changes. As an example, let us consider the representation of a date or a time. Different formats and levels of granularity could be used. The International Standard for the representation of dates and times, the ISO 8601, describes just six levels of granularity and a large number of formats [37]. Different systems could use different formats, thus giving different interpretations of a same date.

In order to be able to define the business services that are made up by the various components on the ESB, it is necessary to be able to specify the required flow from component to component. But the path used to physically get from one component to another has to be understood by the ESB, and indeed the ESB may well want to dynamically alter this path, such as a reaction to a failure in part of the network. The *ContentBasedRouting* component provides a routing service that can intelligently consider the content of the information being passed from one step to another and choose the appropriate next business step based on that information.

It is worth noticing that the ESB provides a general-purpose technical integration infrastructure, and it should not be used for semantic interoperability issues. Indeed, technical integration refers to the design of technological artifacts

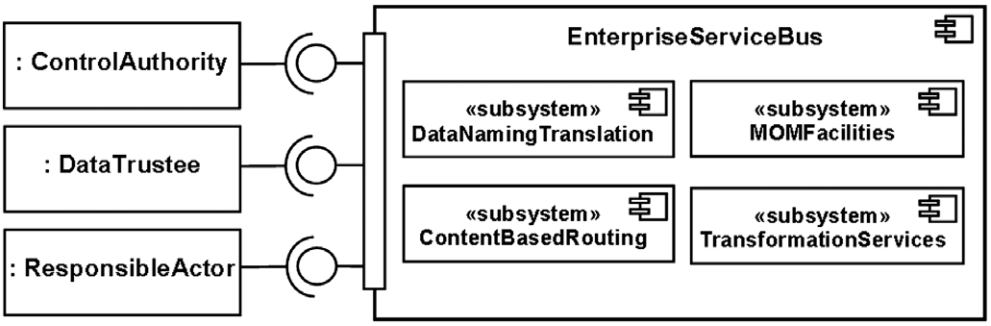

Fig. 10. Generic components of an Enterprise Service Bus for traceability purposes.

that are easy to use as part of a larger suite of components, tools, or services. Semantic interoperability concerns the design of the business interfaces that enable separately designed business processes to be plugged together so they can be flexibly used. How do we define standards that are sufficiently flexible and well defined that they can form the basis for broad integration of heterogeneous, legacy and future systems for traceability issues? Since the traced product history describes relations of commercial partners (a set of sales and purchases), we can refer to business-to-business (B2B) context where interoperability issues have been studied from several years. Next section is devoted to discuss this aspect.

7. Business process interoperability

In the B2B context, EDI has been used for almost a quarter of a century, as a fast and reliable means of achieving electronic, computer-to-computer information exchange between trading partners. The efficacy of using EDI has been widely investigated (see e.g., [38]), and it is now evident that the potential of this standard can be fully exploited by enterprises with mature IT capabilities: often this is not the case for all the actors throughout the supply chain. On the other hand, the proliferation of XML-based business interchanges has served as the catalyst for defining a new global paradigm that ensured all business activities, regardless of size, could engage in electronic business activities. Such a paradigm, known as ebXML [17], has been developed by an international initiative established by the United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT) [39] and the Organization for the Advancement of Structured Information Standards (OASIS) [40]. It enables enterprises to conduct business over the Internet in more straightforward and efficient ways than in the past [41–43]. ebXML represents a set of modular business collaboration-oriented specifications, defined as a set of layered extensions to the base SOAP and SOAP Messages with Attachments (SOAPAttach) specifications [34]. ebXML has been recently standardized, and it is currently known also as ISO 15000 [44]. Business collaboration requires a solid and consistent conceptual foundation, encompassing both issues in inter-enterprise business collaboration (based on mutually accepted trading partner agreements), and a technical infrastructure that (i) enables businesses to find each other and (ii) provides for the reliable and secure exchange of business messages between partners. As well as the plain web services technology, ebXML provides technical interoperability through a vendor-neutral protocol. Moreover, ebXML uses Collaboration Protocol Agreements (CPAs) to declare bindings to business collaboration specifications. ebXML requires collaborating partners to mutually agree upon the formats and semantics of business documents. However, it is not mandatory to exploit only XML-encoded messages in an ebXML-based system: theoretically, EDI messages could be used as well.

In an inter-enterprise business collaboration scenario, both business partners would use the ebXML Message Service (ebMS) to transport business documents in a secure, reliable, and recoverable way. Obviously, in case one of the business partners cannot manage ebMS messages (for instance, in the case of legacy systems), the communication is handled via ESB. Business level failures are completely taken into account with the Business Process Specification Schema (BPSS) [45]. For example, if a party fails to respond within a pre-defined time period, then the BPSS reverts to the previously known secure state.

ebMS is just concerned about message transport: an additional standard is needed in order to define the semantics of a business document (i.e., the message payload). As there are several horizontal and vertical content standards in existence, a novel initiative, called *Universal Business Language* (UBL) [46], is trying to achieve a universal XML business language over ebXML. The message-exchange agreement between two business partners is described by means of a CPA. Any successive change on the interface of a business service identified in the CPA will consequently invalidate the CPA, thus requiring a new formal agreement document to be built. Of course, this kind of modifications does not affect the technical message exchange function. Hence, the sender can still be sure that the message gets delivered, even if the recipient will be likely to experience potential problems with the new message content semantics.

The foundation layer of the ebXML stack, i.e., the XML Schema [47] and the SOAP standards, constitute a

stateless, one-way message exchange paradigm, providing a basic messaging framework for higher abstract layers. Another important set of layered standards is the WS stack [48]. SOAP-WS architecture is the most common and marketed form of web service in the industry. In B2B scenarios, the specific strengths of ebXML and WS can be combined in that ebXML is used for managing enterprise-spanning business transaction services in the context of collaborative business, while WS find their place in intra-enterprise integration of back-end systems. The most important vendors of WS architectures support SOAP, WSDL, UDDI [49] as the primary standards to develop simple WS, and ebXML as the standard for complex WS [50–52].

Actually, both standards implement the principles behind the next generation of e-business architectures [16,53], representing the logical evolution from Object-Oriented architecture (OOA) to systems of services, built according to SOA [54]. The fundamental layer of WS, however, does not consider business process semantics as ebXML does. Not surprisingly, an ESB can be considered as a SOA approach to integration, even though, as we want to emphasize in this section, interoperability, rather than integration, is the most innovative principle of SOA.

SOA promotes significant decoupling and dynamic binding of components: all software components are structured as services, i.e., they encapsulate behavior and expose it to other collaborating components on the network by means of standard messaging facilities. In the SOA approach, applications are built by discovering and orchestrating network-available services, or by just-in-time integration of applications. Fig. 11 represents the SOA pattern using a UML component diagram [55]. Here, *Provider* and *Requester* are interpreted as roles of components implementing and using an interface, respectively. The *Broker* service acts as a registry that implements a special interface to publish/query service descriptions by providers/requesters.

The three fundamental operations in this kind of architecture are *publishing*, *finding*, and *binding*. Service providers publish services to a service broker, service requesters find required services using the service broker and bind to them. In the current version of ebXML specifications, like in UDDI, the service broker is implemented by a Registry. The ebXML Registry exposes services for information sharing between parties. The typical SOA-compliant use of the ebXML registry can be sketched by describing the following scenario, shown in Fig. 12 as a UML communication diagram.

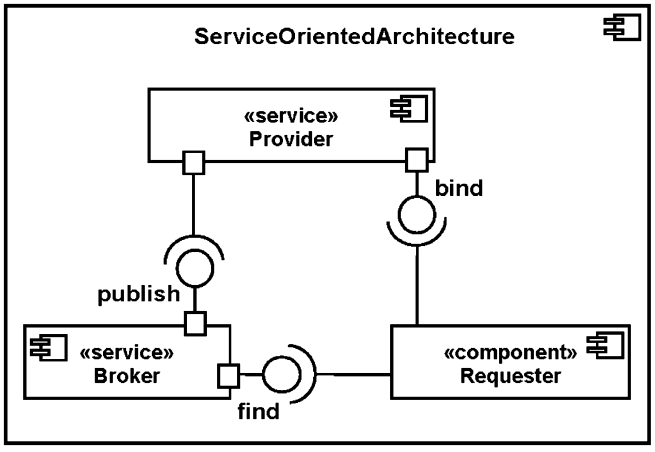

Fig. 11. Generic components of a Service Oriented Architecture.

First, a company *A* becomes aware of an ebXML Registry. Second, company *A*, after reviewing the contents of the ebXML Registry, decides to build and deploy its own ebXML compliant application. Third, company *A* then submits its own Business Profile information to the ebXML Registry, describing its capabilities and constraints, as well as its supported business scenarios. Fourth, a company *B* discovers these business scenarios in the ebXML Registry, and (fifth) sends a request to company *A* to engage in one of them. In particular, company *B* acquires an ebXML compliant application and submits a proposed business arrangement directly to the software interface of company *A*. Sixth, companies *A* and *B* are ready to engage in e-business using ebXML.

The ebXML Registry standard defines both the data to be held in the registry/repository, and the service interfaces to access such data. In addition, the standard has been designed extensible so as to allow any artifact to be incorporated. ebXML has also defined how registries can be federated in such a way that a SOA solution can be developed across multiple registries; this is an essential function because the intent of SOA is that services may be provided by multiple suppliers (either different organizations) within an enterprise or external organizations. The definition of an artifact must be owned and controlled by the producer, and therefore must be in a registry/repository controlled by the producer. Therefore multiple registries will be involved in any significant SOA implementation and namely also in traceability systems built on ebXML technologies.

8. Traceability in the food supply chain: the Cerere project

The scenarios described in the previous sections have been established in a project named "Cerere". The methodology followed in the project is made up of three main and overlapping activities, namely *setup*, *build-up*, and *test-up* activities. The *setup activity* consists of three sub-tasks: (i) analysis of the business processes and transactions inherent in traceability, (ii) development of detailed specifications and design of data schemas so as to support automatic application-to-application transactions, and (iii) drafting of detailed specifications for the technological infrastructure of the overall system. The *build-up activity* is devoted to the implementation of the messaging/repositories servers and to the development of a set of reference client applications, which allow the interconnection of different systems. Finally, the vertical *test-up activity* encompasses the three following sub-tasks: (i) trial and pilot operation of the

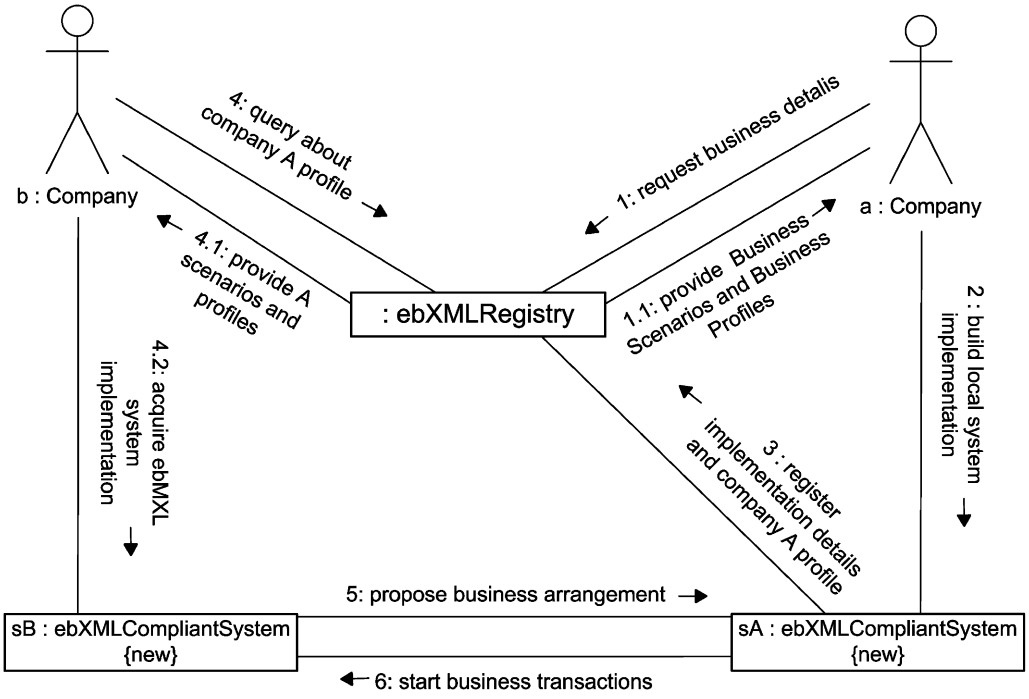

Fig. 12. A high level UML communication diagram of the interaction of two companies conducting e-business using ebXML.

entire system with select groups of users, (ii) validation of the system for a variety of products and users and evaluation of its capability to adapt itself to different contexts, and (iii) the dissemination and exploitation of the project results.

As regards the first two activities, the modeling of the production processes for a given supply chain is accomplished according to the following procedure. First, the traceability technical disciplinary (for the production processes) and the supply chain management regulation (for the administrative processes) are analyzed. Second, the business processes extracted from the disciplinary are modeled through UML and translated into XML documents, as required by the BPSS specification.

The application of the methodology has allowed the Cerere project to achieve the following results:

1. the design of a set of business processes, protocols, and reference architecture for the interoperability of information systems in food supply chains;
2. the development of a software prototype that allows the smart interconnection of such systems and the safe asynchronous transfer of transaction data between businesses or other legal entities;
3. some pilot studies carried out in four real supply chains corresponding to four different contexts: olive oil, wine, bread-making, and vegetable.

In our methodological context, ebXML is used as the reference specification to define and exchange business documents in the interoperability architecture. In particular, in the Cerere project we decided to adopt the Hermes Message Service Handler (MSH) [56], because of its full support to ebMS, message packaging and ordering, reliability, error handling, security, synchronous reply, message status service, and RDBMS persistent storage. Another important feature is the employment of CPAs among collaborating actors, required by our project and provided by MSH. The business documents (including the CPAs) involved in the Cerere framework are handled by means of an open-source implementation of ebXML Registry (the freebXML Registry-OMAR [57]). To edit the ebXML documents, the freebXML BP visual editor has been used [58]. To edit the specific context XML business documents, a general-purpose visual XML editor has been developed, with a lightweight Java applet web interface. It is worth noticing that the use of a visual editor is of fundamental help in the accessibility of the aforementioned documents, which determine the actual tuning of the traceability platform used to execute simulations.

An important methodological step in the ebXML modeling is the mapping between UML and XML entities. The definition of the XML Schema documents exchanged via Hermes MSH has been accomplished by conforming to the following rule, coming from established best practices: if UML objects and attributes are structured ones, or they can assume a large set of values, then they are translated into XML *elements*; otherwise, they are mapped onto XML *attributes*. More specifically, 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*. Fig. 13 shows the basic structure of three XML documents for acquisition, providing,

a

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

b

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

c

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

Fig. 13. Three simplified XML documents used for acquisition (a), providing (b), and transformation (c) notification.

and transformation notification in a distributed setting (we refer to the entities in Figs. 3 and 5). Since in a distributed architecture the exchanged data amount must be kept as small as possible, we can notice that the ebXML messages just include the lot and actor identifiers.

Fig. 14 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 features are shown in Fig. 15, for numerical and categorical types, respectively.

The simplified software infrastructure is shown in Fig. 16, using a UML deployment diagram. It is worth pinpointing that the main component of the message switching system (i.e., MSH) takes care of validating Cerere documents, as well as sending/receiving them over HTTP. Furthermore, it provides error-handling facilities for a number of situations that may arise in real life. The final customer can access traceability information through a Web Interface connected to the system backend and leveraging a WS infrastructure. Whenever Process Collaboration is the main goal, the MSH System provides the proper interaction protocol among peers, according to a document-centric approach [59]. On the other hand, the access to Business Information Services can be proficiently achieved via WS, able to provide an efficient and lightweight RPC-based interaction.

The architectural design process leads to a clear and well-structured organization of the modules involved in the traceability system. This has allowed addressing the typical issues for this broad category of information systems. We would like also to underline that each specific domain covered by actual traceability systems presents particular challenges in obtaining the required performance level. In traceability systems, because of their typical asynchronous nature, no strict Quality of Service requirement is usually present for the interaction among actors in the

a

```
<activity>
  <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>
```

b

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

Fig. 14. A simplified XML document instance of an activity (a) and a lot (b).

a

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

b

```
<qualityFeature>
    <description>rating</description>
    <categoricalQF>
        <value>two stars</value>
        <categoricalValue
            value ="one star"
            ordering ="0"
            description ="good"/>
        <categoricalValue
            value ="two stars"
            ordering ="1"
            description ="very good"/>
        <categoricalValue
             value ="three stars"
             ordering ="2"
             description ="excellent"/>
    </categoricalQF>
</qualityFeature>
```

Fig. 15. An XML translation for numerical (a) and categorical (b) quality features.

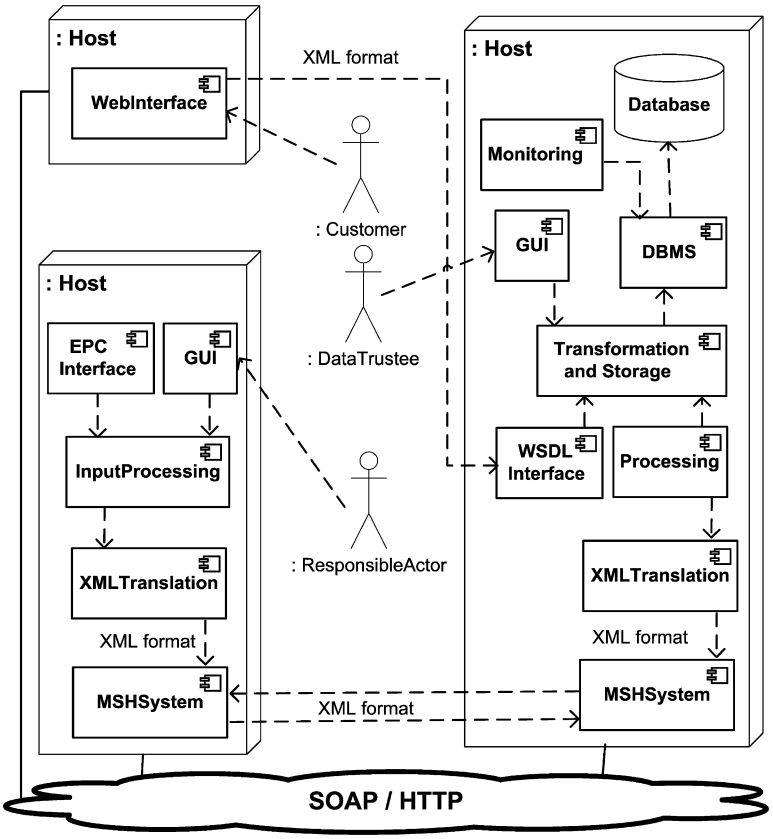

Fig. 16. The simplified communication architecture for the Cerere system, shown by a UML deployment diagram.

supply chain. For this reason, performance is not the main concern in the presented architecture. Nevertheless, particular attention must be paid to performance whenever scalability becomes crucial to the system operation. The overall design has been arranged to isolate potential performance issues within a single component (namely, the messaging module), so that possible required improvements have to involve just the design, implementation and configuration of the messaging subsystem. Although XML processing for B2B documents has been recognized as a computing-intensive activity [53,60], today the efficiency improvements obtained by the employment of opti-

mized libraries for this kind of tasks can substantially mitigate potential performance problems from the XML-processing modules.

9. Conclusions

Design and implementation of traceability systems need preliminary investigations to point out problems and solutions at different abstraction levels. The foundation for any possible discussion about the development of this kind of systems is represented by the adoption of a generic data model for traceability. We have proposed such a model, expressed using UML, describing its basic classes and the patterns used to represent the lot behavior along the supply chain.

In order to achieve data integration along the supply chain for traceability purposes, a common, widely accepted set of specifications for collaboration is required. In this context, XML and SOAP can be surely regarded as emerging enabling technologies. The plain help from XML and SOAP is not sufficient to address all the semantic aspects of each document exchange; document communication and sharing among business partners should be unambiguously modeled. A recently proposed standard to provide semantics elements and properties necessary to define business collaborations is ebXML: its employment in the context of traceability has been discussed in this paper. The goal of the ebXML specification is to provide the bridge between e-business process modeling and specification of e-business software components. Business process models describe interoperable business processes that allow business partners to collaborate. The use of ebXML-related solutions has been weighted up, taking integration and interoperability requirements into account. The evaluation of these standards has been discussed in the framework to the SOA paradigm.

Under the Cerere project, we have developed a prototype of a traceability system and a corresponding Web Information System for the food supply chain, which is able both to trace and to track product units and batches. The system takes also information about product quality into account. Such a system has been applied to four real food supply chains, composed of a wide assortment of small and medium enterprises. The participating enterprises play different roles in the supply chain and are characterized by different levels in technological competence, economic resources, and human skills. In this setting, the ebXML technology has been successfully adopted to support collaboration among all the involved actors at different stages of the supply chain. The supply chain processes are defined by means of CPA documents and messages are delivered to a data trustee repository using ebXML mechanisms.

The benefits for the partners that have adopted the Cerere system can be summarized in the following aspects. First, a significant reduction of the time and effort needed to execute every-day transactions, in terms of several person-months per year. Second, a significant reduction in rate of errors that are currently caused by replicated data entries and manual interventions. Third, a reduced cost in the adoption of e-business processes, since the Cerere system has been made widely available and freely distributed.

Finally, we would like to point out important open problems, which are related to the automated negotiation process. This process is controlled by the CPA, starts with the negotiation of variables from two CPPs and moves upward to the negotiation of application domain entities. In this context, several difficulties have been experienced in automating the definitions of business terms, conditions and parameters, related to the agribusiness sector. In order to overcome these difficulties, we have created a common XML vocabulary for specific agribusiness documents, to embed specialized business logic in a collection of business document templates. In fact, generic ontologies (as UBL) are not directly usable in the traceability domain, because they do not contain domain-specific concept definitions. Thus, we realized that, though traceability fundamentals are independent of types of products, and production and control systems, the task of building traceability ontologies is very time-expensive, since it is partially composed of manual tasks performed by skilled actors. To this aim, a methodology is currently under development, and a validation procedure has been planned, so as to make such methodology reusable as reference in different agribusiness contexts.

Acknowledgement

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

References

[1] L.U. Opara, Traceability in agriculture and food supply chain: a review of basic concepts, technological implications, and future prospects, European Journal of Operational Research 159 (2004) 269–295.
[2] GS1 Traceability, http://www.gs1.org/productssolutions/traceability, 2006.
[3] M.H. Jansen-Vullers, C.A. van Dorp, A.J.M. Beulens, Managing traceability information in manufacture, International Journal of Information Management, Elsevier 23 (5) (2003) 395–413.
[4] Regulation (EC) n. 178/2002 of the European Parliament and of the Council of January 28, 2002, Ch. V.
[5] U.S. Food and Drug Administration, regulation 21CFR820, Title 21: Food and drugs, subchapter H: Medical devices, part 820 Quality system regulation, http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfcfr/CFRSearch.cfm?CFRPart=820, 2004.
[6] Ministry of Agriculture, Forestry and Fisheries of Japan, Guidelines for Introduction of Food Traceability Systems, http://www.maff.go.jp/trace/guide_en.pdf, 2003.
[7] Food Standards Agency of United Kingdom, Traceability in the Food Chain – A preliminary study, http://www.foodstandards.gov.uk/news/newsarchive/traceability, 2002.
[8] 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: Proceedings of EFITA Conference, Debrecen, Hungary, 2003, pp. 607–611.

[9] C.A. van Dorp, Tracking and tracing business cases: incidents, accidents and opportunities, in: Proceedings of EFITA Conference, Debrecen, Hungary, 2003, pp. 601–606.
[10] J. Hagel, A.G. Armstrong, Net Gain: Expanding Markets Through Virtual Communities, Harvard Business School Press, Boston, MA, USA, 1997.
[11] B. Roberts, A. Svirskas, B. Matthews, Request Based Virtual Organisations (RBVO): an Implementation Scenario, in: Collaborative Networks and Their Breeding Environments: Sixth IFIP Working Conference on Virtual Enterprises, (PRO-VE'05), in: L.M. Camarinha-Matos, H. Afsarmanesh, A. Ortiz (Eds.) IFIP, Springer-Verlag, New York, NY, USA, 2005, pp. 17–24.
[12] V. Choudhury, Strategic choices in the development of inter-organizational information systems, Information Systems Research 8 (1) (1997) 1–24.
[13] T.W. Malone, J. Yates, R.I. Benjamin, Electronic markets and electronic hierarchies: effects of information technologies on market structure and corporate strategies, Communications of the ACM 30 (6) (1987) 484–497.
[14] J.P. Silva, G.D. Putnik, M.M. Cunha, Technologies for virtual enterprise integration, in: Business Excellence: Performance measures, Benchmarking and Best Practices in New Economy, University of Minho Press, 2003, pp. 706–712.
[15] UN/EDIFACT – United Nations Directories for Electronic Data Interchange for Administration, Commerce, Transport, http://www.unece.org/trade/untdid/welcome.htm, 2006.
[16] A. Sahai, S. Graupner, Web Services in the Enterprise: Concepts, Standards, Solutions, and Management, Springer, New York, NY, USA, 2005.
[17] B. Hofreiter, C. Huemer, W. Klas, ebXML: status, research issues, and obstacles, in: Proceedings of the Twelfth IEEE Research Issues in Data Engineering: Engineering E-Commerce/E-Business Systems (RIDE'02), San Jose, CA, USA, 2002, pp. 7–16.
[18] A. Gunasekaran, E.W.T. Ngai, Information systems in supply chain integration and management, European Journal of Operational Research 159 (2) (2004) 269–295.
[19] H.M. Kim, M.S. Fox, M. Gruninger, Ontology of quality for enterprise modelling, in: Proceedings of IEEE International Workshop on Enabling Technologies: Infrastuctures for Collaborative Enterprises, Los Alamitos, CA, USA, 1995, pp. 105–116.
[20] J. Rumbaugh, I. Jacobson, G. Booch, The Unified Modeling Language Reference Manual, second ed., Addison Wesley, Boston, MA, USA, 2004.
[21] GS1 official website, http://www.gs1.org/, 2006.
[22] ISO 9000 family of standards, http://www.iso.org/iso/en/iso9000-14000/index.html, 2006.
[23] W.R. Pape, B. Jorgenson, T. Horsley, R.D. Boyle, In data collection the first mile is the toughest, Food Traceability Report, 2004, pp. 14–15.
[24] R.M. Hornby, RFID solutions for the express parcel and airline baggage industry, in: Proceedings IEE Colloquium on RFID Technology, London, UK, 1999.
[25] G. Roussos, Enabling RFID in retail, IEEE Computer 39 (3) (2006) 25–30.
[26] EPCglobal official website, http://www.epcglobalinc.org/, 2006.
[27] Food Standards Agency of United Kingdom, Guidance on the Implementation of Articles 11, 1 and 16–20 of Regulation (EC) No. 178/2002 on General Food Law, UK, 2004.
[28] W.R. Pape, B. Jorgenson, R.D. Boyle, J. Pauwels, Let's get animal traceback right the first time, Food Traceability Report, 2004, pp. 14–15.
[29] W.R. Pape, B. Jorgenson, R.D. Boyle, J. Pauwels, Selecting the most appropriate database architecture, Food Traceability Report, 2003 pp. 21–23.
[30] B. Meng, Q. Xiong, Research on electronic payment model, in: Proceedings of the International Conference on Computer Supported Cooperative Work in Design (CSCWD'04), Xiamen, China, 2004, pp. 597–602.
[31] M. de Mello, P. de Azevedo, Effects of Traceability on the Brazilian Beef Agribusiness System, Virtual Library Osvaldo "Bebe" Silva, Quilmes, Argentina, 2000.
[32] J. Hagel, J.S. Brown, The Only Sustainable Edge, Harvard Business School Press, Boston, MA, USA, 2005.
[33] XML, eXtensible Markup Language, http://www.w3.org/XML/, 2006.
[34] SOAP, http://www.w3.org/TR/soap/, 2006.
[35] D.A. Chappel, Enterprise Service Bus, O'Really, 2004.
[36] A.Y. Halevy, Z.G. Ives, D. Suciu, I. Tatarinov, Schema mediation for large-scale semantic data sharing, The International Journal on Very Large Data Bases 14 (1) (2005) 68–83.
[37] ISO 8601, Numeric representation of Dates and Time, http://www.iso.org/iso/en/prods-services/popstds/datesandtime.html, 2006.
[38] T.T. Pham, Managerial considerations to adopting electronic data interchange, in: Proceedings of IEEE International Conference on E-Commerce Technology (CEC'03), Newport Beach, CA, USA, 2003, pp. 136–145.
[39] UN/CEFACT – United Nations Centre for Trade Facilitation and Electronic Business, http://www.unece.org/cefact/, 2006.
[40] OASIS – Organization for the Advancement of Structured Information Standards, http://www.oasis-open.org, 2006.
[41] B. Gibbs, S. Damodaran, ebXML Concepts and Application, John Wiley & Sons, New York, NY, USA, 2002.
[42] P.F. Green, M. Rosemann, M. Indulska, Ontological evaluation of enterprise systems interoperability using ebXML, IEEE Transaction on Knowledge and Data Engineering 17 (5) (2005) 713–725.
[43] A. Kotok, D.R.R. Webber, ebXML: The New Global Standard for Doing Business on the Internet, New Riders Publishing, Thousand Oaks, CA, USA, 2001.
[44] ISO official website, http://www.iso.org/iso/en/CatalogueListPage.CatalogueList, 2006.
[45] ebXML Business Process Specification Schema v1.01, http://www.ebxml.org/specs, 2006.
[46] UBL specification v1.0 – http://www.oasis-open.org/committees/ubl, 2006.
[47] C. Binstock, D. Peterson, M. Smith, M. Wooding, C. Dix, C. Galtenberg, The XML Schema Complete Reference, Addison Wesley Professional, Boston, MA, USA, 2003.
[48] Web Services Architecture, http://www.w3.org/TR/ws-arch/, 2004.
[49] OASIS, Universal Description, Discovery and Integration, http://www.uddi.org/, 2006.
[50] Oracle9i Application Server, Web Services Strategy, http://www.oracle.com/technology/products/ias/web_services/pdf/webservicesstrategy_twp.pdf, 2001.
[51] Service-Oriented Architecture, Sun's Service Registry, http://www.sun.com/products/soa/registry/, 2006.
[52] Web Services by IBM, http://www-06.ibm.com/software/solutions/webservices/casestudies/xmlglobal.html, 2006.
[53] K. Hogg, P. Chilcott, M. Nolan, B. Srinivasan, An Evaluation of Web Services in the design of a B2B application, in: Proceedings of the 27th Australasian Conference on Computer Science, Dunedin, New Zealand, vol. 26, 2004, pp. 331–340.
[54] S. Jones, Toward an acceptable definition of service, IEEE Software 22 (3) (2005) 87–93.
[55] R. Heckel, M. Lohmann, S. Thone, Towards a UML profile for service-oriented architectures, in: Proceedings of Workshop on Model Driven Architecture: Foundations and Applications, Twente, Enschede, The Netherlands, 2003, pp. 115–120.
[56] Hermes Message Service Handler – http://www.cecid.hku.hk/hermes.php, 2006.
[57] ebXMLRR website, http://ebxmlrr.sourceforge.net/index.html, 2006.
[58] freeXML BP Editor website, http://www.freebxml.org/freebXMLbp.htm, 2006.
[59] D.E. Jenz, "ebXML and Web Services – Friends or Foes?", http://www.mywebservices.org/index.php/article/articleview/451/1/1/, 2002.
[60] C. Kohlhoff, R. Steele, Evaluating SOAP for high performance business applications: real-time trading systems, in: Proceedings of World Wide Web Conference (WWW'03), Budapest, Hungary, 2003.