Overview

• Introductive traceability scenario
• Interaction among participants
• Business Processes interoperability (ebXML)
• Overall architecture of a prototype
• Future work
Product recall scenario

Goal:
• efficient recall

Terms:
• Traceability
• Tracking
• Tracing
• Lot
• Activity
• Relation

Quality

Contamination Incident

uncontaminated units

potentially contaminated units
Interaction among participants

- Each responsible actor belongs to a different company
- Information exchanges among responsible actors
- Heterogeneous structure and naming of Data
- Tackling heterogeneous semantics
- Confidentiality and control of data. Intermediate data trustees
- Large and Dynamic community (depending on the market)
- Managing the Business status of agreements
- Facing failure scenarios
Interaction among participants

- (a) distributed and (b) central management
- Identifiers attached to the physical lot
- “Push” strategy
Interaction among participants

- Distributed management, intermediate data trustee
- “Pull” strategy
Early technological needs:

- Highly distributed architecture
- Dealing with multiple software interfaces
- Tackling etherogeneity
- Loosely coupling communication

Early methodological needs:

- Relying on standard inter-organizations cooperation models and protocols
- Strongly separating the Business level from the technical one.
BP interoperability

Traceability and e-business standards:

- **ebXML** (Electronic Business using eXtensible Markup Language)

- Globally developed standard (ISO15000) started in 1999 as an initiative of OASIS and the United Nations/ECE agency CEFACT.
BP interoperability

ebXML specifications:

- Technical Architecture (TA)
- Message Services (ebMS)
- Collaboration Protocol Agreements / Collaboration Protocol Profile (CPA / CPP)
- Business Process Specification Schema (BPSS)
BP interoperability

ebXML outline:

• Many trading partners collaborate together to create working relationship
• Interchange defined as requestor / responder
• Business transactions exchanged control the state of the process
• Sharing of definitions and business understanding between partners within a community
• Support for Business Scalability (smaller companies can participate, not just large corporations)
BP interoperability

ebXML overview: basic semantics of a business collaboration (BPSS)
BP interoperability

ebXML overview: Relationship of ebXML BPSS to UMM, CPP/CPA and Core Components
ebXML Messaging Service (ebMS) implementation

Center for E-Commerce Infrastructure Development. University of Hong Kong

Main features:
- message packaging
- reliable messaging
- message ordering
- error handling
- security
- synchronous reply
- message status service
- persistent storage
- QoS support (CPA)

Basic Message Service Handler Architecture
Overall architecture of a prototype

Outline of a traceability framework
Overall architecture of a prototype

Generic traceability semantics

- two packages
- traceable entity
- traceable identifier: EAN/UCC (barcode), EPC (RFID)
Overall architecture of a prototype

Generic traceability semantics

- **Lot** → Acquisition → Lot → Providing → Lot
- **Lot** → Division → Lot
- **Lot** → Alteration → Lot
- **Lot** → Transformation
- **Lot** → Integration → Lot

Diagram components:
- **Lot**
- **Acquisition**
- **Providing**
- **Division**
- **Alteration**
- **Transformation**
- **Integration**

Steps:
- a)
- b)
- c)
- d)
- e)
- f)
Overall architecture of a prototype

Lot state diagram (choreography)

- At the beginning, a lot is acquired by an actor from the "Nature".
- A lot is transformed by an activity into another lot.
- At the end, an actor provides the "Nature" or the "Consumer" with a lot.
Overall architecture of a prototype

Involved activities (choreography)

```
Nature
[extracted]

[damaged]

Responsible Actor

Acquisition

Lot [acquired]

Transformation

Lot [transformed]

Providing

Lot [provided]

Consumer

[consumed]
```
Overall architecture of a prototype

Simplified cheese supply chain


Nature  Supplier  Shop  Consumer

1: acquisition()  1: create()  3: transformation()  3: create()  5: acquisition()  6: providing()  6: create()

2: transformation()  2: create()  4: providing()  4: create()  5: create()  5: create()  5: create()
Overall architecture of a prototype

XML translations (for a purchase activity)

```
<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) activity

<b) lot

<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>
```
Conclusions and Future Work

- Logical view (BPSS)
- Message Orientation (ebMS)
- Description Orientation (UML, XML)
- Network Orientation (SOAP/HTTP)
- Platform Neutral (XML)

- We are currently experiencing the application of the prototype to a real vegetable supply chain.

- The employment of Collaboration Protocol Agreements and Registries implementations should be taken in account.