"Le grandi e complesse organizzazioni aziendali sono la manifestazione tangibile della tecnologia avanzata, più delle stesse macchine."[1]

INTRODUZIONE AD EBXML

caso di studio: *la tracciabilità di filiera*

Mario G.C.A. Cimino

[1] J. K. Galbraith, "The New Industrial State", 1967.

Concetti introduttivi sui Sistemi Informativi

- ✓ La citazione di cui sopra fu scritta quando "elettronica per uffici" significava un citofono tra il capo ufficio e la segretaria, e il "sistema di informazione aziendale" era costituito da bacheche e matite. Negli anni successivi si sarebbero sviluppati i **centri di elaborazione dati**, per l'elaborazione semi-automatica di documenti, ed i **sistemi di automazione industriale**.

- ✓ Solo a partire dal 1985, dalla integrazione di questi due modelli nacque e si sviluppò sino al 2000, il modello dei **sistemi informativi**, un sistema di persone, procedure e strumenti per svolgere processi gestionali elaborando informazioni e fornendo un supporto integrato a livello di intera organizzazione.

- ✓ Negli ultimi tempi (2000) il modello di **e-business** ha introdotto l'idea del controllo automatizzato di tutti i processi aziendali attraverso cui monitorare ogni tipo di attività grazie alla infrastuttura pervasiva fornita dalla rete Internet. Oggi, le maggiori organizzazioni stanno rivedendo i propri **processi di business** in base ai futuri sviluppi e alle potenzialità di Internet.

- ✓ Per **business** si intende un'attività di tipo *(i)* industriale oppure *(ii)* commerciale. Per **processo di business** l'unità di suddivisione funzionale del business, ossia qualsiasi metodo o sistema per *(i)* fare un prodotto/servizio oppure *(ii)* acquistarlo/venderlo. Es. approvigionamenti, pagamenti, spedizioni.

- ✓ Il sistema informativo di un'azienda dipende dalla organizzazione aziendale. La descrizione accurata dei processi aziendali è alla base per la realizzazione dei **sistemi informatici**. Consideriamo il seguente esempio[2].

> *Verso la metà degli anni settanta, la Aamco, una grossa azienda commerciale per la vendita all'ingrosso di ricambi automobilistici, decise di diversificarsi aprendo una catena di negozi di articoli da regalo chiamata Plum Tree. Il progetto prevedeva l'utilizzo dello stesso sistema centralizzato di acquisto e rivendita in concessione che si era rivelato vantaggioso per i pezzi di ricambio: comprare in grande quantità costa meno, e le concessionarie avrebbero potuto acquistare i singoli articoli di cui avevano bisogno. Collegando l'acquisto centralizzato a un accurato monitoraggio delle vendite di ogni articolo, il sistema poteva tenere basse le scorte di magazzino mantenendo allo stesso tempo le riserve necessarie a sostenere le vendite. Dopo un avvio promettente, le vendite della Plum Tree dapprima si assestarono e poi iniziarono a declinare fino all'uscita definitiva dal settore. Il sistema di controllo degli ordini e dell'inventario che aveva funzionato così bene per i ricambi automobilistici fallì nel caso di articoli da regalo. Il sistema ordinava automaticamente una maggiore quantità dei ricambi usati con più frequenza, allo scopo di mantenere il flusso dei pezzi nelle officine di riparazione e di ottenere un prezzo più competitivo per gli articoli più venduti. E, identicamente, ordinava un maggior numero degli articoli da regalo che i clienti compravano più frequentemente. Ma i ricambi automobilistici e gli articoli da regalo non vengono richiesti allo stesso modo, sebbene possano essere descritti dai medesimi attributi (quantità, prezzo,…). Un meccanico normalmente ripete lo stesso tipo di riparazione giorno dopo giorno, quindi avrà bisogno ogni volta degli stessi pezzi. Chi acquista articoli da regalo, invece, difficilmente comprerà ogni volta lo stesso oggetto. Vedendosi offrire soltanto il "ricambio" di ciò che era già stato presentato, i clienti smisero di comprare.*

[2] A.Penzias , "Ideas and Information: managing in a high-tech world", 1989.

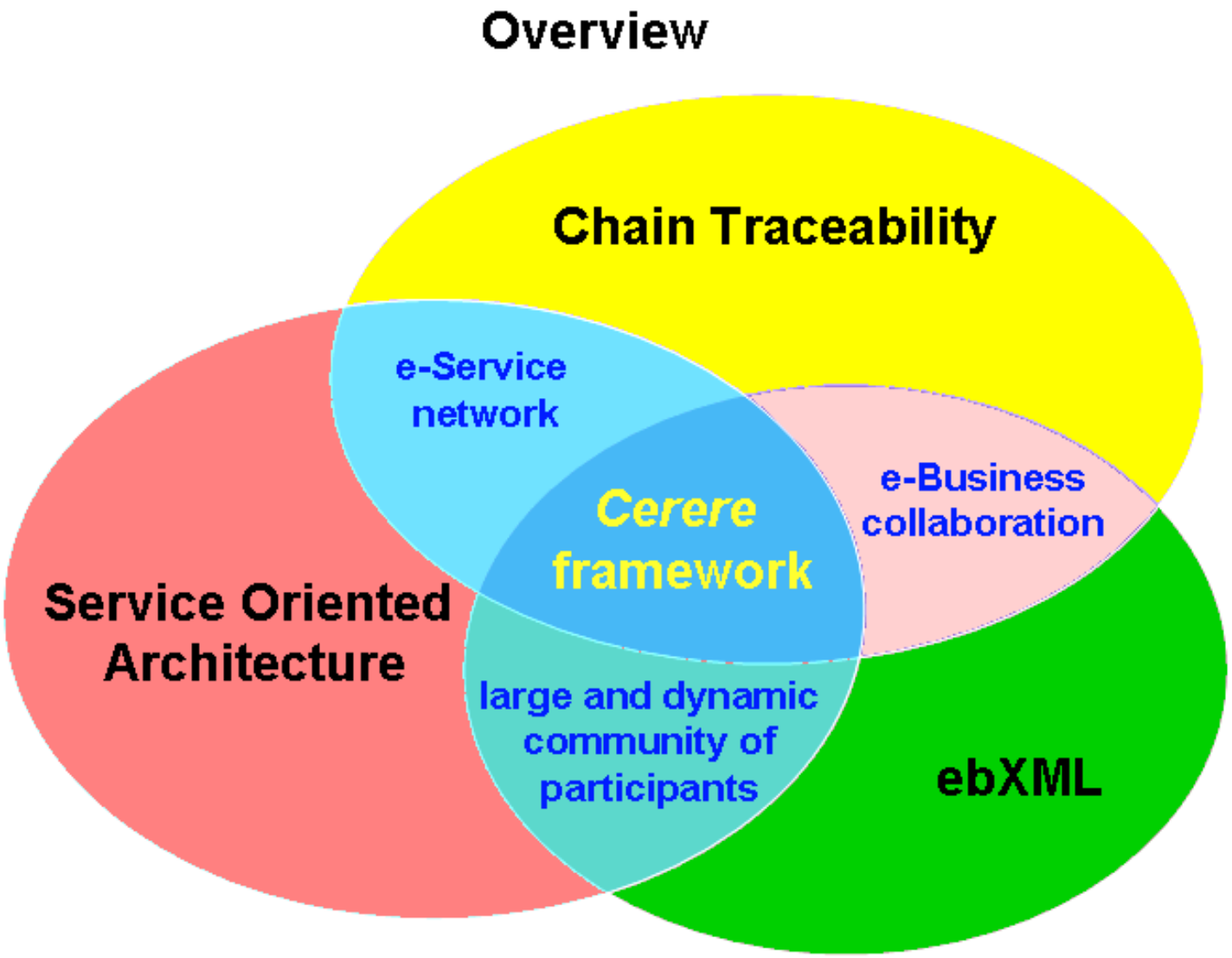
- ✓ Un sistema informatico, per essere "plasmato" dai processi aziendali (e non il viceversa) deve essere innanzitutto **modulare**, (ciascun modulo è finalizzato al supporto di specifiche categorie di processi), **estendibile** (si possono aggiungere nuove funzionalità) e caratterizzato da altissimo livello di **integrazione** (può connettersi a diverse piattaforme per il trasferimento di dati e servizi) e di **interoperabilità** (può co-operare con diverse piattaforme adeguandosi a differenti semantiche di business).

- ✓ I processi aziendali sono in pratica infinitamente scomponibili in attività più elementari, per cui occorre seguire dei **pattern di tipo gestionale** a cui riportarsi per analizzarli ed identificarli. Il progettista del sistema informatico traduce le attività, modellate come servizi a livello di business, in sottosistemi interagenti secondo **pattern di tipo tecnologico**.

- ✓ Da quanto esposto, si evidenzia una struttura gerarchica delle strategie, di seguito riassunta e raffigurata: gli obiettivi di un'organizzazione sono raggiungibili attraverso i propri processi di business, i quali richiedono risorse (materiali, persone, attrezzature, capitali) e tecnologie, che vanno acquisite, sviluppate ed organizzate.


strategie
processi aziendali
risorse e tecnologie
amministrazione e organizzazione
livello strategico
livello business
livello tecnologico
livello operativo

Legami tra Tracciabilità, SOA ed ebXML



- ✓ La tracciabilità è un servizio strategico nella supply chain: può essere usata per migliorare la sicurezza, controllare la qualità, combattere le frodi ed amministrare filiere complesse.

- ✓ Dal punto di vista degli utenti, le organizzazioni nella filiera necessitano di un metodo standard per definire e registrare processi di business, scambiare messaggi di business, condurre relazioni di commercio, comunicare dati in termini comuni. In tale direzione, ebXML offre un insieme di standard per i processi di business, componenti di business riusabili, protocolli di accordo, messaggistica, registri ed archivi.

- ✓ Dal punto di vista dei progettisti, costruire un sistema di tracciabilità coinvolge tutti gli stadi di produzione, trasformazione e distribuzione. Quindi occorre uno stile architetturale che garantisca l'interoperabilità e la portabilità dei componenti software in rete. In questa prospettiva, SOA è l'unico paradigma che assicura una cooperazione effettiva tra agenti software eterogenei.

- ✓ In realtà, sulla base dei concetti SOA, ebXML è in grado di gestire una vasta comunità di partecipanti, amministrare risorse centrali per consentire ai partner di unirsi rapidamente nella comunità ed integrare le loro proprie applicazioni in tale rete di servizi.

SOA e Core Web Services

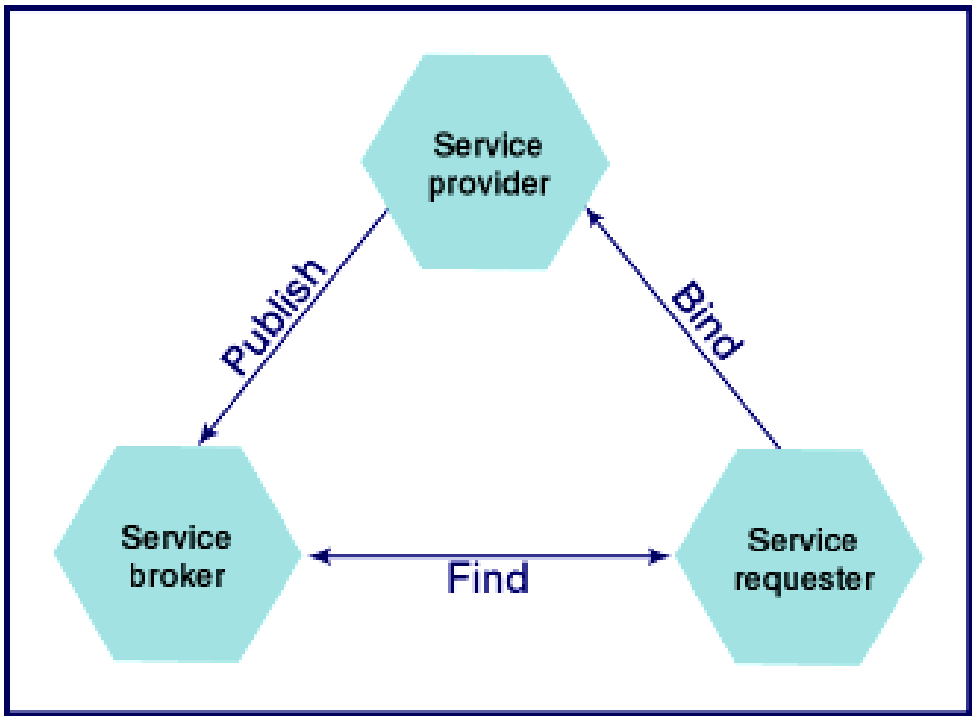
- ✓ Negli ultimi anni le metodologie per la modellazione di architetture informatiche si sono standardizzate nel paradigma Object Oriented (**OO**), convergendo in un linguaggio unificato, Unified Modeling Language (**UML**), adoperato anche nella modellazione dei processi di business.

- ✓ **SOA** è un paradigma di progettazione di recente introduzione, per certi versi alternativo ad OO. Si basa sul concetto di servizio (Service Oriented Architecture) e propone uno schema in cui agenti software, con un certo grado di autonomia strutturale ed operativa, cooperano dinamicamente sul web offrendo e cercando servizi di qualsiasi natura.

- ✓ Un *service provider* è un nodo che fornisce una interfaccia per svolgere un insieme di compiti. Esso rappresenta i servizi di una entità di business o semplicemente l'interfaccia di servizio per un sottosistema riusabile.

- ✓ Un *service requester* è un nodo della rete in grado di scoprire ed invocare altri servizi software per fornire una soluzione di business. Spesso è un componente di un'applicazione di business che esegue chiamate di procedura remote ad un oggetto distribuito, il *service provider*.

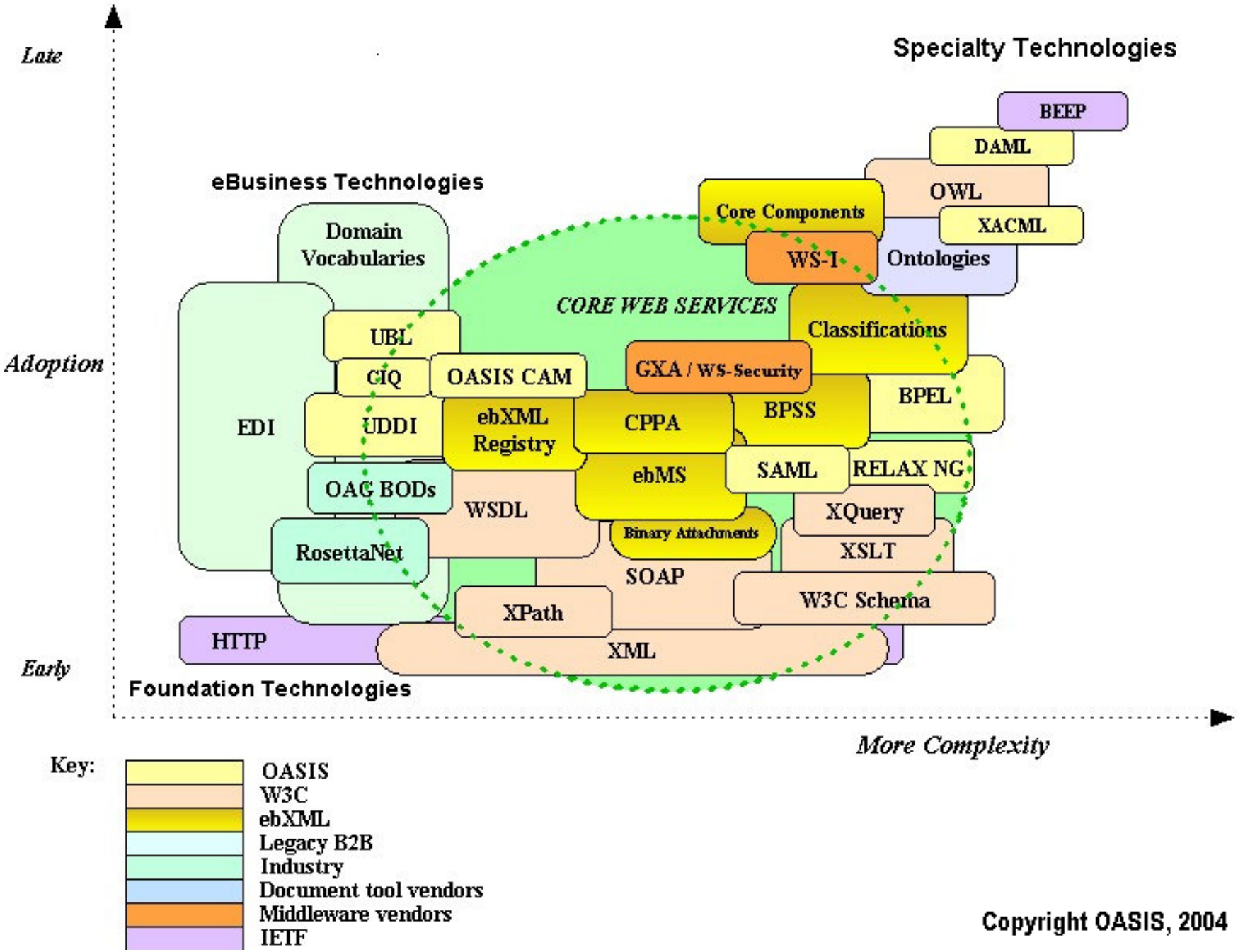
- ✓ Il *service broker* è un nodo che funge da repository o "pagine gialle", per interfacce software che sono pubblicate dai *service provider*. Può essere un'entità di business oppure un operatore indipendente.



- ✓ I tre partecipanti SOA interagiscono mediante le tre operazioni base: publish, find and bind. I service provider *pubblicano* i servizi su un service broker. I service requester *trovano* i servizi richiesti usando un service broker e si *legano* ad essi.

- ✓ Un tale grado di interoperabilità richiede un metalinguaggio standard ed una tecnologia pervasiva che consenta di raggiungere i servizi in un contesto globale.

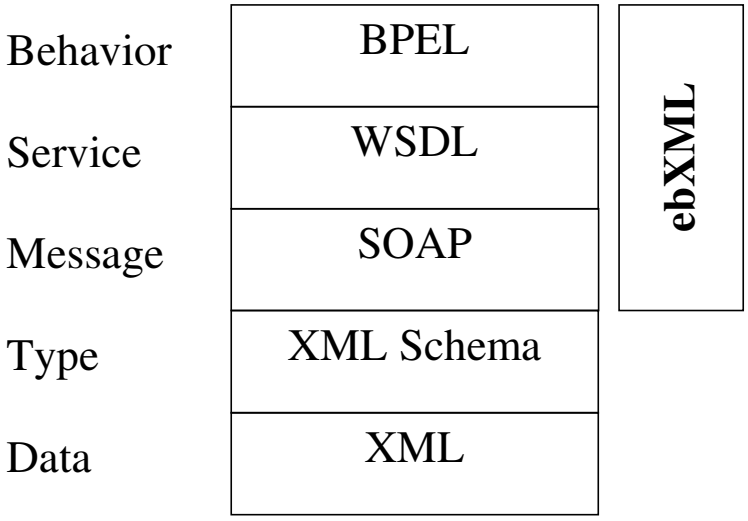
- ✓ Proprio il modello di **e-business** ha portato alla formulazione di un metalinguaggio (eXtensible Markup language, **XML**) per la definizione di grammatiche sui diversi contesti di business e l'introduzione di framework tecnologici quali Web Services (**WS**) e-business using XML (**ebXML**) che consentono di esprimere gli aspetti semantici ed operativi del business mascherando il substrato tecnologico ed adoperando il web come supporto.


Late
Adoption
Early
Specialty Technologies
BEEP
DAML
OWL
XACML
Ontologies
eBusiness Technologies
Core Components
WS-I
Domain Vocabularies
CORE WEB SERVICES
Classifications
UBL
GXA / WS-Security
CIQ
OASIS CAM
BPEL
BPSS
EDI
UDDI
ebXML Registry
CPPA
SAML
RELAX NG
ebMS
OAG BODs
WSDL
XQuery
Binary Attachments
RosettaNet
XSLT
SOAP
W3C Schema
XPath
HTTP
XML
Foundation Technologies
More Complexity
Key:
OASIS
W3C
ebXML
Legacy B2B
Industry
Document tool vendors
Middleware vendors
IETF
Copyright OASIS, 2004

ebXML e WS

- ✓ Tecnicamente parlando, due organizzazioni che intendono cooperare in rete hanno bisogno di accordarsi su come invocare i rispettivi servizi/processi di business e come scambiare dati. Entrambi hanno bisogno di avere a comune protocolli, formati e contenuti dei messaggi.
- ✓ Le *business collaboration* si possono racchiudere in tre categorie:
  - *Business information services*: le organizzazioni condividono informazioni, es. le quotazioni dei titoli o le news.
  - *Business integration services*: una organizzazione fornisce servizi integrativi ai clienti es. sistemi di prenotazione, di controllo del credito.
  - *Business transaction services*: due organizzazioni si impegnano in una operazione di business mutuamente vincolante, con obblighi chiaramente definiti, es. acquisto di prodotti, contrattazione di servizi di trasporto. Questa tipologia di servizi è spesso legata a processi di business.
- ✓ Si dice che le organizzazioni cooperano con "legame debole" se nessuna di esse può esercitare controllo sull'altra (es. un acquirente non deve poter bloccare le risorse di un venditore).
- ✓ Per trovare i servizi occorrono dei registri pubblici. Infine occorre stabilire le conseguenze in caso di condizioni di malfunzionamento (es. problemi di comunicazione).

- ✓ *ebXML* si colloca su un livello parallelo rispetto allo stack di tecnologie standard dei *Web Services (WS)*. Entrambi forniscono interoperabilità tecnica attraverso un protocollo non proprietario, ma sono complementari perchè si occupano di diverse categorie di business collaboration.
- ✓ I *WS* sono tipicamente appropriati per *e-business non collaborativo* ossia *business information services* e *business integration services*, mentre i *business transaction service* sono adeguatamente supportati da *ebXML*.



Web Service Standards

SOAP, Simple Object Access Protocol, insieme minimale di convenzioni per lo scambio di dati interpiattaforma, codificati in XML e trasferiti mediante HTTP.

WSDL, Web Service Definition Language, una grammatica XML per descrivere un servizio web come una collezione di punti di accesso in grado di scambiare messaggi secondo un paradigma a *procedure* o a *documenti*.

BPEL, Business Process Execution Language, linguaggio XML-based per definire una composizione di servizi nella forma di coreografie di WS, ossia di un'aggregazione di WS secondo regole di interazione prestabilite. Questo standard consente di formulare i WS come business process

- ✓ L'uso di un WS non richiede un accordo tra richiedente e fornitore, mentre le *business collaboration* previste da ebXML prevedono un *Collaboration Protocol Agreement (CPA)* che definisce i servizi messi a disposizione dalle due controparti

(una sorta di interfaccia) sulle quali deve trovarsi un accordo prima di impegnarsi nelle transazioni.

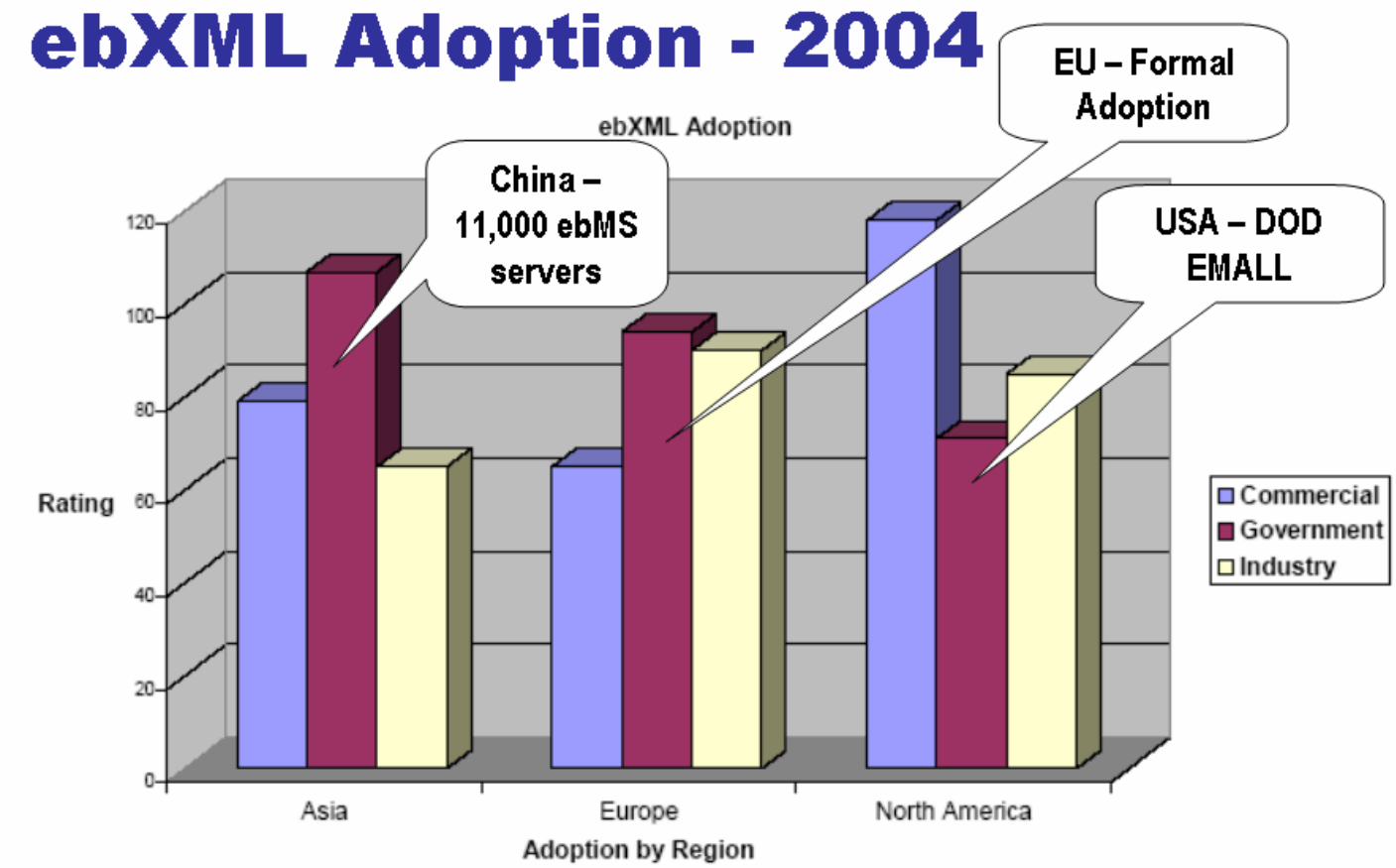
	Web Services	**ebXML**
Type	Request/response	Collaboration
Communication	RPC-style synchronous communication between tightly coupled services, Document-style asynchronous communication between loosely coupled services	Synchronous, asynchronous communication
Business Service Interface description	WSDL	CPP, CPA (WSDL within CPP, CPA under research)
Protocol and Formats	SOAP, XML	ebXML Message Service (over SOAP), XML, BPSS (as "business" protocol)
Content Standards	None	Recommended Standards (e.g. OAGI BODs)
How to find business partners	UDDI Registry	ebXML Registry (UDDI Registry may point to an ebXML Registry or Registry objects (e.g. CPA))

UDDI, Universal Description, Discovery and Integration, metodo standardizzato, XML-based, per pubblicare e ritrovare informazioni sui WS

OAGI (Open Applications Group, Inc.) ha sviluppato un vasto insieme di messaggi di business (messaggi scambiati fra le parti, in ebXML) e scenari di integrazione per enterprise application e B2B, senza specificare una architettura di implementazione (implementation framework). ebXML fornisce uno standard per trasportare OAGI **BODs** (Business Object Documents).

- ✓ A differenza dell'interfaccia nel WSDL, un cambio dell'interfaccia nel CPA non influenza lo scambio tecnico dei messaggi, ma invalida solo il CPA. Inoltre, in ebXML anche i problemi a livello di business sono presi in considerazione, nel *Business Process Specification Schema (BPSS)*. Ad esempio, se una parte non risponde entro un predefinito periodo di tempo, il BPSS ritorna allo stato precedente.

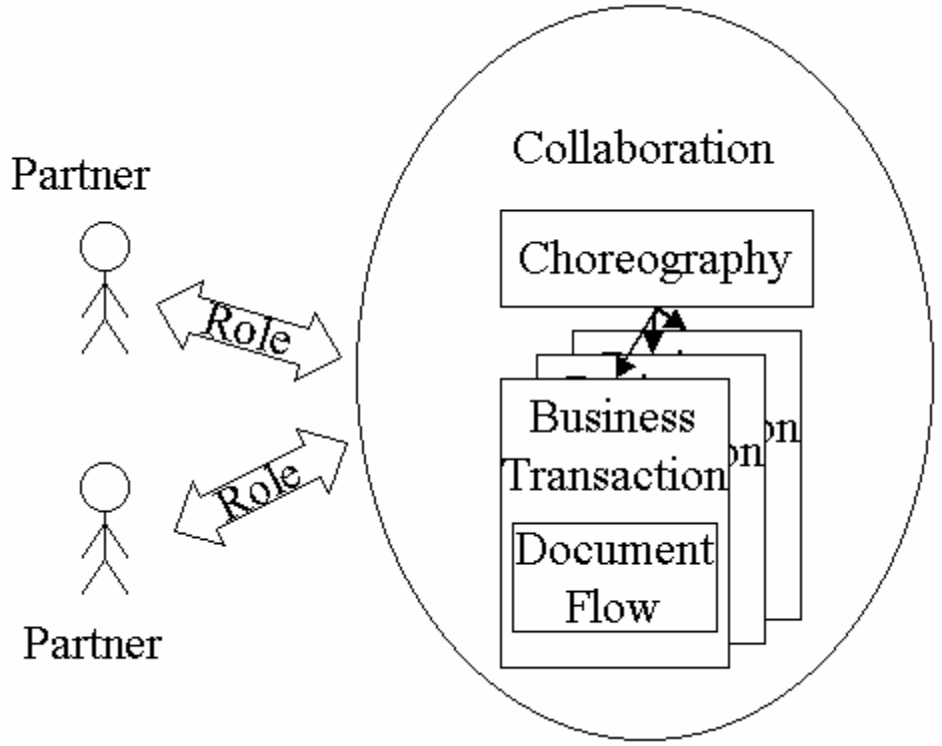
- ✓ Negli scenari B2B, ebXML trova consenso per amministrare *enterprise-spanning business transaction services* nel contesto di business collaborativo; invece WS trovano posto in *intra-enterprise integration of back-end systems*.
- ✓ In termini di standardizzazione, ebXML è in fase più avanzata di WS. Il primo fornisce una credibile collaborazione di business ed un'automazione di processi inter-enterprise, il secondo fornisce SOAP, WSDL ed UDDI.



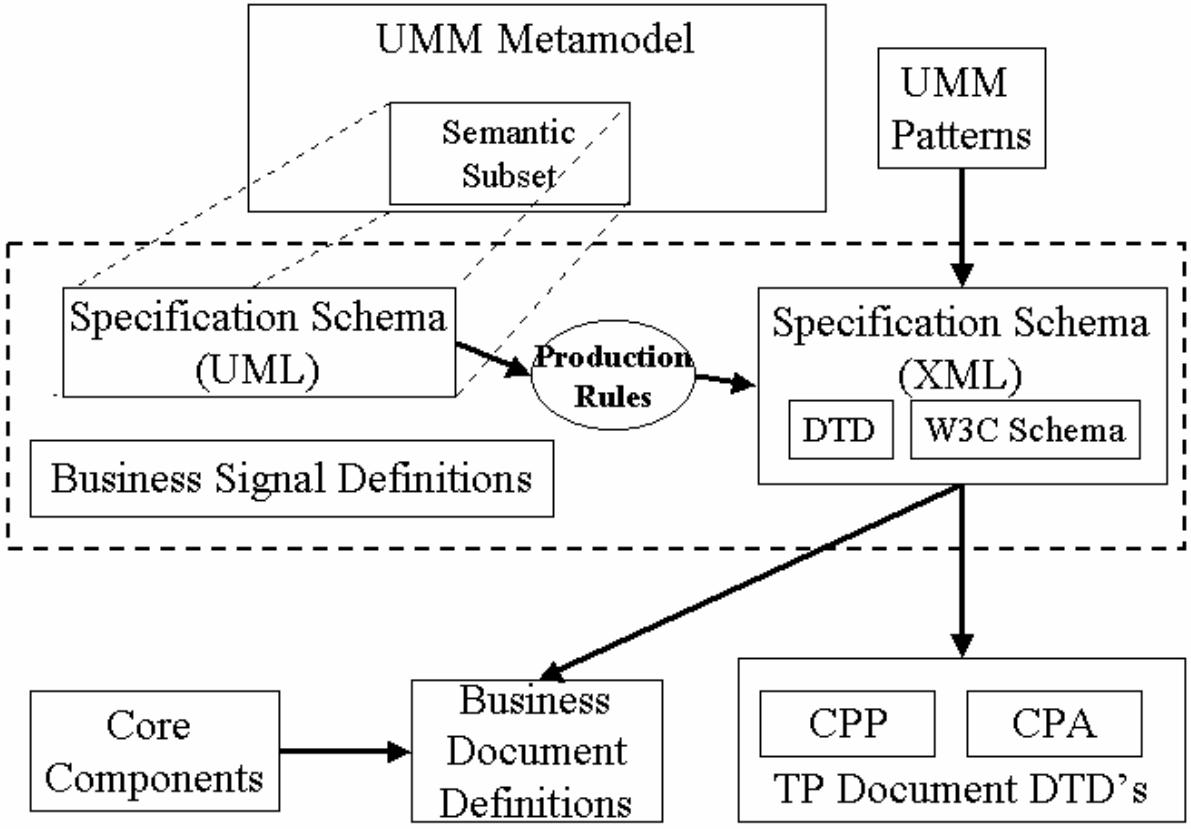
Comparison of ebXML adoption by region, number / size / scope of projects as of December 22, 2003

Entità fondamentali di ebXML

- ✓ *Business process*, la definizione di un *business process model* riguarda i "run time aspects" ossia l'ordine con cui vengono inviati i messaggi in una *business collaboration*, senza considerare la elaborazione dei dati, e definisce i *ruoli* dei partecipanti, cui corrispondono transazioni in un ordinamento stabilito dalla *business transaction coreograhy.* Una *business transaction* consiste in una fase atomica di comunicazione (con un meccanismo di *rollback* in caso di fallimento) in cui avviene lo scambio di documenti.



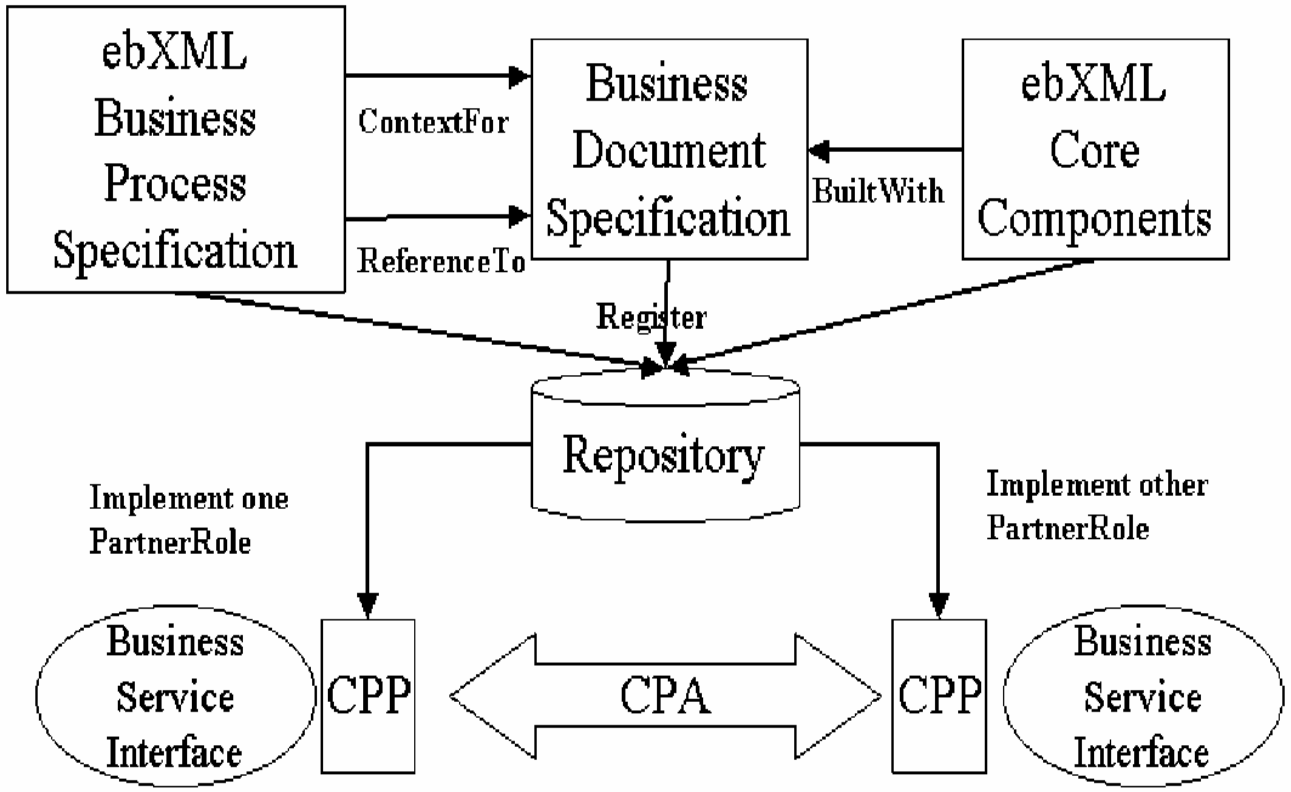
- ✓ **BPSS**, *Business Process Specification Schema* definisce un *business process model* attraverso (i) un diagramma delle classi UML, (ii) uno schema XML, (iii) delle regole di produzione da UML ad XML, (iv) le *Business Signal Definitions* (documenti a livello applicativo, distinti dal protocollo di trasporto di basso livello, che segnalano lo stato corrente delle transazioni)



- ✓ **UMM**, *UN/CEFACT Modeling Methodology*, metodologia per i processi di business e la modellazione consistente dell'informazione. UMM Meta Model è una

descrizione di semantiche di business che permette a partner commerciali di inquadrare i dettagli per uno specifico scenario di business.

- ✓ I Business Document sono composti da Business Information Documents riusabili. A basso livello, i Business Information Objects sono composti da **Core Components** riusabili.
- ✓ **CPP**, *Collaboration Protocol Profile*, definisce le potenzialità sia tecnologiche (protocolli di messaggistica e di comunicazione supportati) sia commerciali (quali Business Collaboration supporta) ed i modi in cui una parte può impegnarsi in business elettronico con altre parti.
- ✓ **CPA**, *Collaboration Protocol Agreement*, definisce l'accordo tra le parti, ed è creato mediante elaborazioni e negoziazioni derivanti dall'incrocio di due CPP. Ad esempio: un CPA include solamente quegli elementi che sono comuni o compatibili fra le due parti.



- ✓ ***Business Service Interface***: descrive il modo in cui una società è in grado di eseguire le transazioni necessarie nel suo processo di business. Inoltre include le tipologie di messaggi supportati e i protocolli sopra cui questi viaggiano.
- ✓ ***Registry/****Repository*: è un contenitore pubblico per modelli di processo, vocabolari e profili dei partner. Può essere implementato con un server centrale.

Esempi

```
<BusinessTransaction name="Create Order">
   <RequestingBusinessActivity name=""
      isNonRepudiationRequired="true"
      timeToAcknowledgeReceipt="P2D"
      timeToAcknowledgeAcceptance="P3D">
      <DocumentEnvelope BusinessDocument="Purchase Order"/>
   </RequestingBusinessActivity>
   <RespondingBusinessActivity name=""
      isNonRepudiationRequired="true"
      timeToAcknowledgeReceipt="P5D">
      <DocumentEnvelope isPositiveResponse="true"
         BusinessDocument="PO Acknowledgement"/>
   </RespondingBusinessActivity>
</BusinessTransaction>
```

- ✓ *Esempio di Business Transaction*, in questo esempio ci sono due flussi di documenti e tre segnali di business. Poiché viene richiesto il "non ripudio" la business activity dovrà conservare i business document nella forma originale in modo che l'informazione di ambedue le parti non possa essere disconosciuta. La richiesta richiede sia la ricevuta che l'accettazione, la risposta solo il riconoscimento di accettazione.

- ✓ “P#D” è uno schema W3C adottato per lo standard ISO8601 (per la rappresentazione della data ed ora in un contesto globale) e significa Period = # Days (a partire dall’invio della richiesta)
- ✓ Una business transaction consiste di una Requesting Business Activity, una Responding Business Activity, 1-2 flussi di documenti tra queste, eventualmente associati ad 1-2 Business Signals di riconoscimento dei flussi.
- ✓ Il segnale di failure viene inviato se ad esempio scade uno dei timeout.

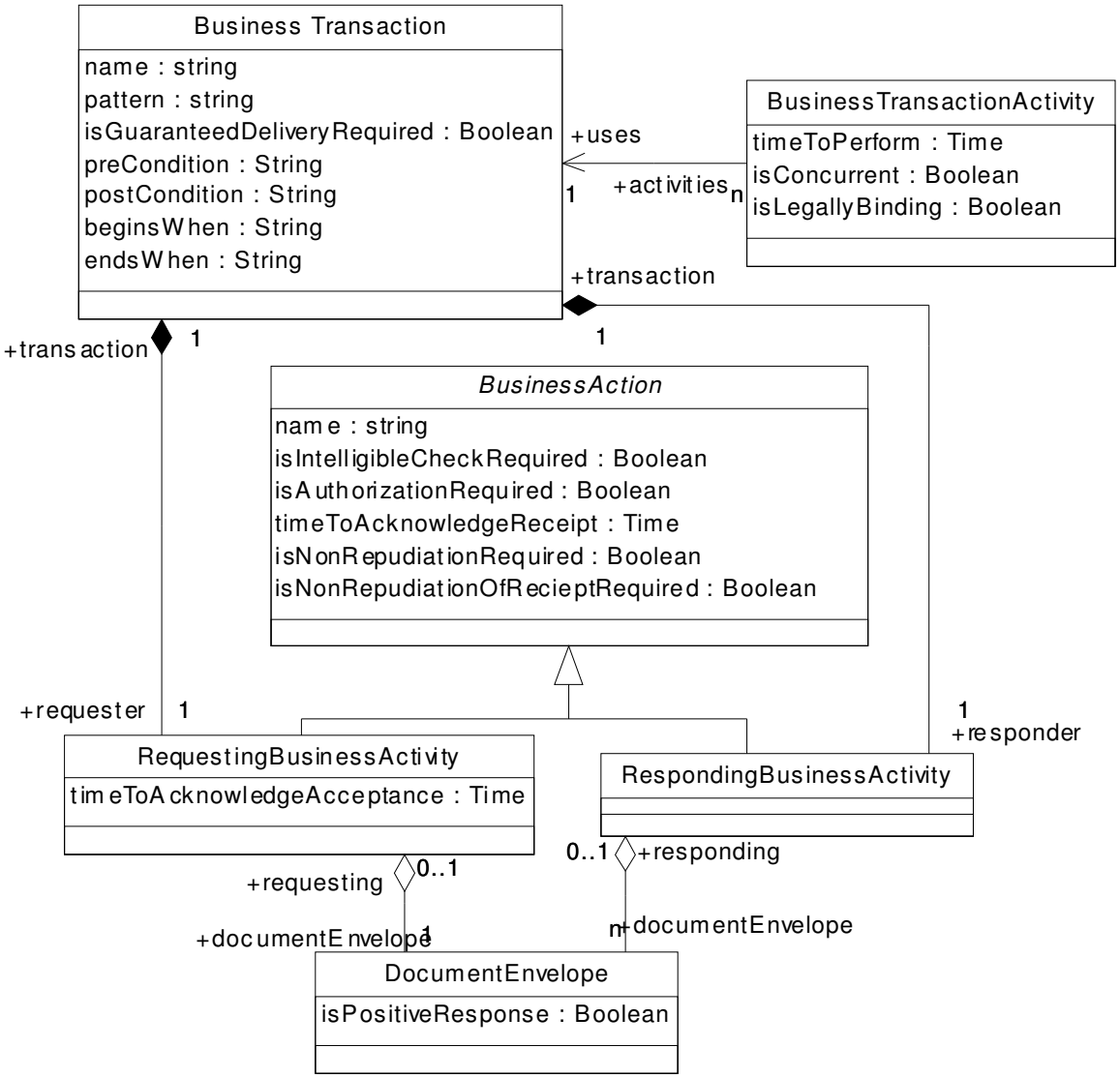
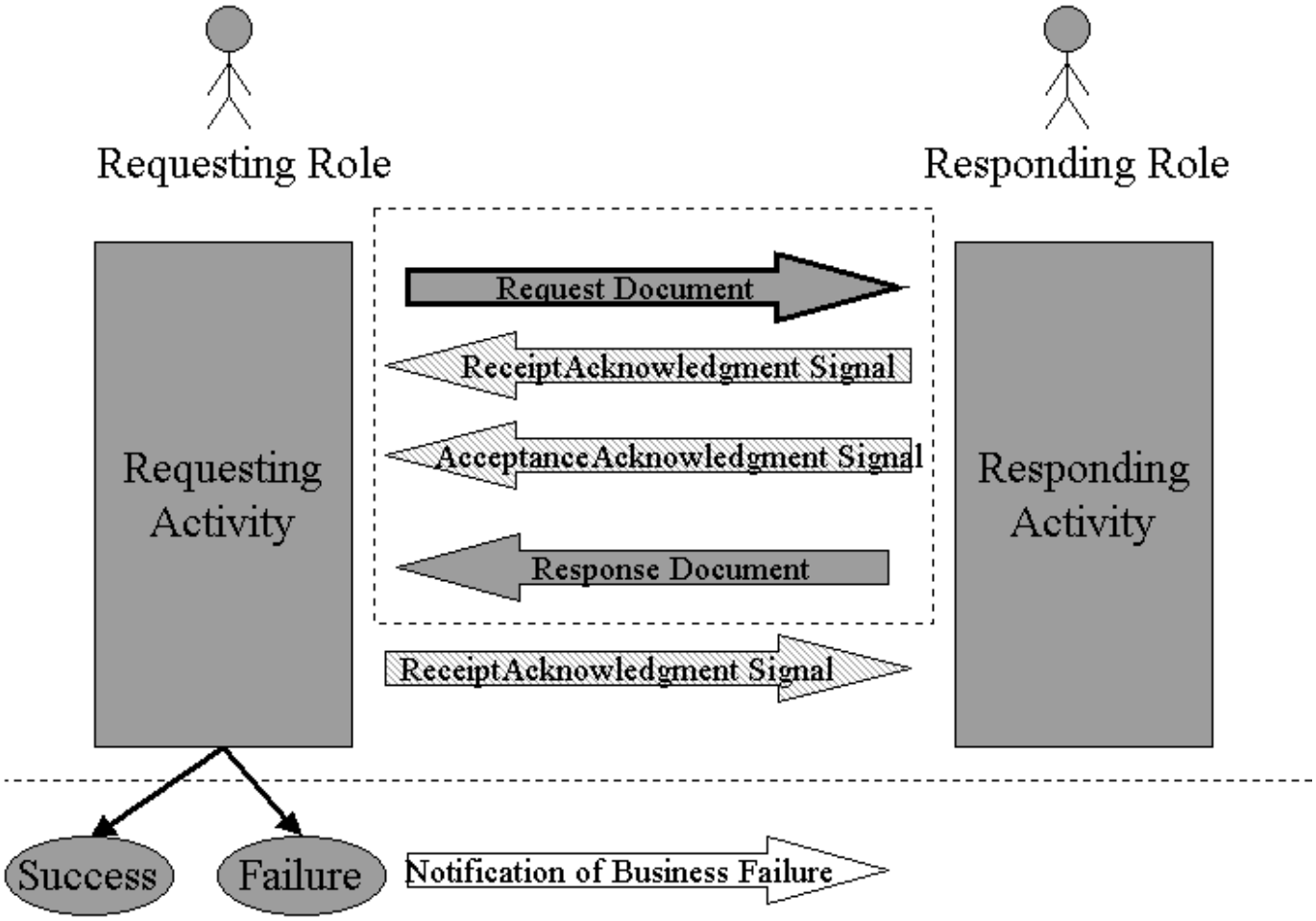


Diagramma UML di una *Business Transaction*



Possibile flusso di documenti e relativa sequenza

- ✓ *Esempio di Flusso di documenti*, sono modellati indirettamente come Involucri di Documenti (Document Envelope) inviati da una parte e ricevuti dall'altra, associati con una richiesta di attività si business ed una corrispondente risposta.

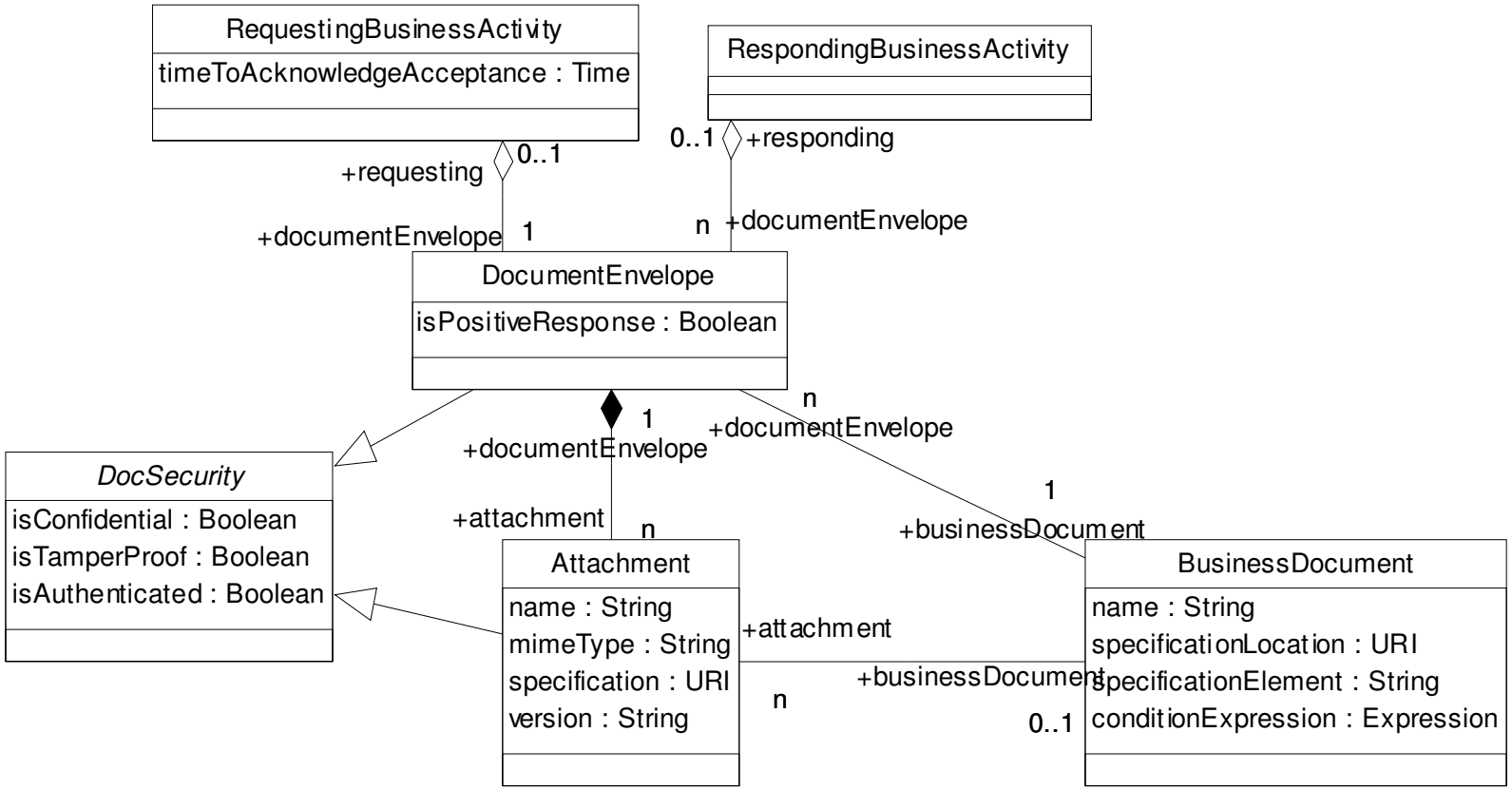


Diagramma UML di un *Document Flow*

- ✓ C'è sempre un solo Document Envelope associato ad una Requesting Activity, questo può avere degli attachment, tutti legati al Business Document primario.

- ✓ In questo esempio la transazione ha una richiesta e due possibili risposte, successo o fallimento. La richiesta ha un allegato. Tutti i business document sono qualificati con lo schema name.

```
<BusinessDocument name="Purchase Order"         specificationLocation="someplace"/>
<BusinessDocument name="PO Acknowledgement"     specificationLocation="someplace"/>
<BusinessDocument name="PO Rejection"           specificationLocation="someplace"/>
<BusinessDocument name="Delivery Instructions" specificationLocation="someplace"/>
<BusinessTransaction name="Create Order">
   <RequestingBusinessActivity name=""
      <DocumentEnvelope isPositiveResponse="true"
         BusinessDocument="ebXML1.0/PO Acknowledgement">
         <Attachment   name=”DeliveryNotes”
                       mimeType=”XML”
                       BusinessDocument="ebXML1.0/Delivery Instructions"
                       specification=””
                       isConfidential=”true”
                       isTamperProof=”true”
                       isAuthenticated=”true”>
         </Attachment>
      </DocumentEnvelope>
   </RequestingBusinessActivity>
   <RespondingBusinessActivity name=""
      <DocumentEnvelope BusinessDocument="ebXML1.0/PO Acknowledgement"/>
      <DocumentEnvelope isPositiveResponse="false"
         BusinessDocument="ebXML1.0/PO Rejection"/>
   </RespondingBusinessActivity>
</BusinessTransaction>
```

- ✓ *Esempio di Collaborazione Binaria*, definisce un protocollo di interazione tra due ruoli autorizzati, coreografata da un insieme di stati di questi ruoli.

timeToPerform	periodo di tempo dall'inizio della prima attività entro il quale l'intera collaborazione deve concludersi
preCondition	descrizione dello stato esterno a questa collaborazione, richiesto prima che essa inizi
postCondition	descrizione dello stato che non esiste prima ma esisterà dopo che questa collaborazione verrà eseguita
beginsWhen / endsWhen	descrizione di un evento esterno alla collaborazione, che normalmente causa l'inizio/fine della medesima.
pattern	riferimento opzionale ad un pattern sul quale questa collaborazione si basa
role	una collaborazione binaria consiste di due ruoli autorizzati, uno di iniziazione ed uno di risposta
states	stati della coll.binaria, statico o stato azione
usedBy	una coll. Binaria può essere usata dentro un'altra c.binaria in una attività di collaborazione
transitions	La transizione tra attività

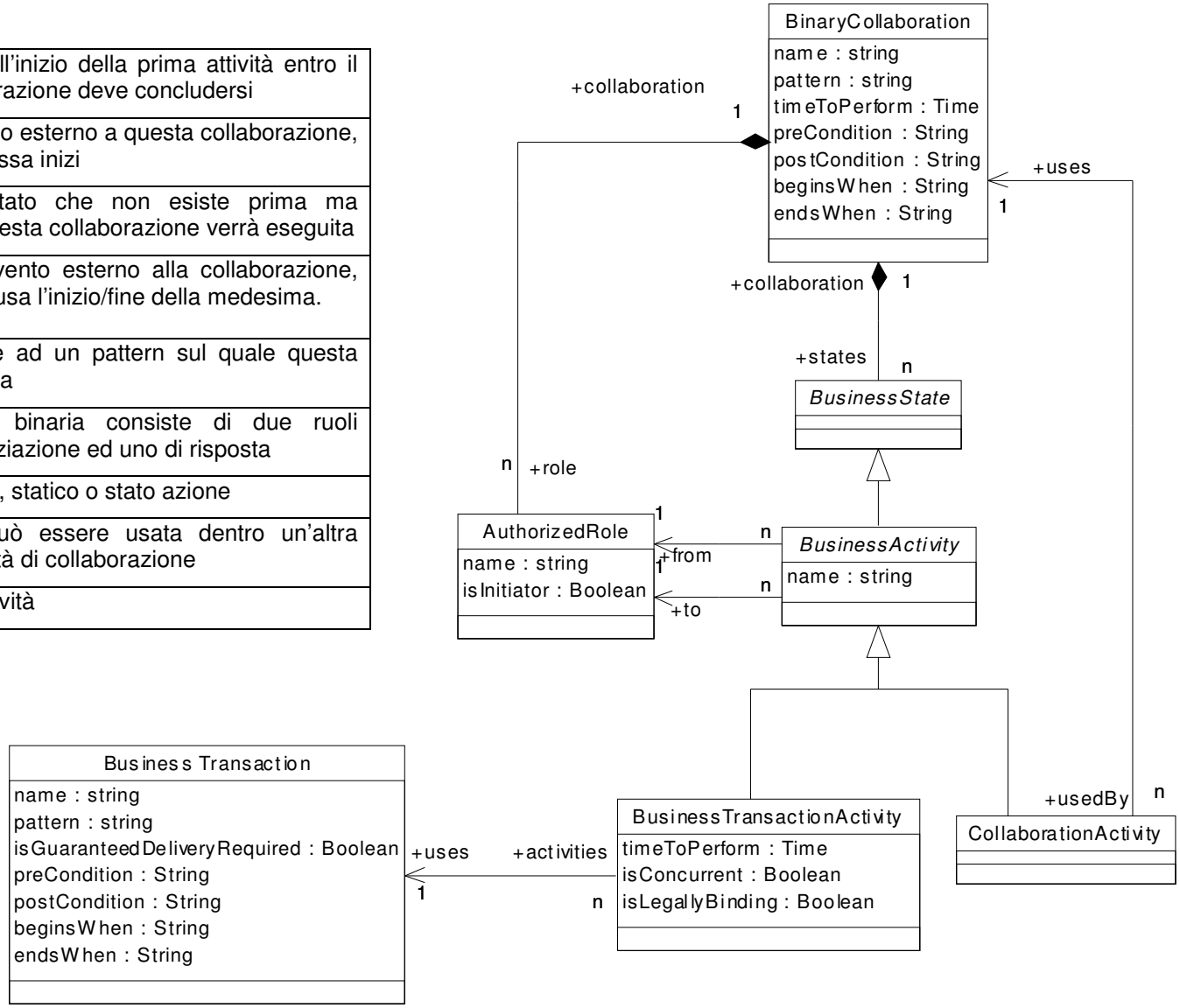
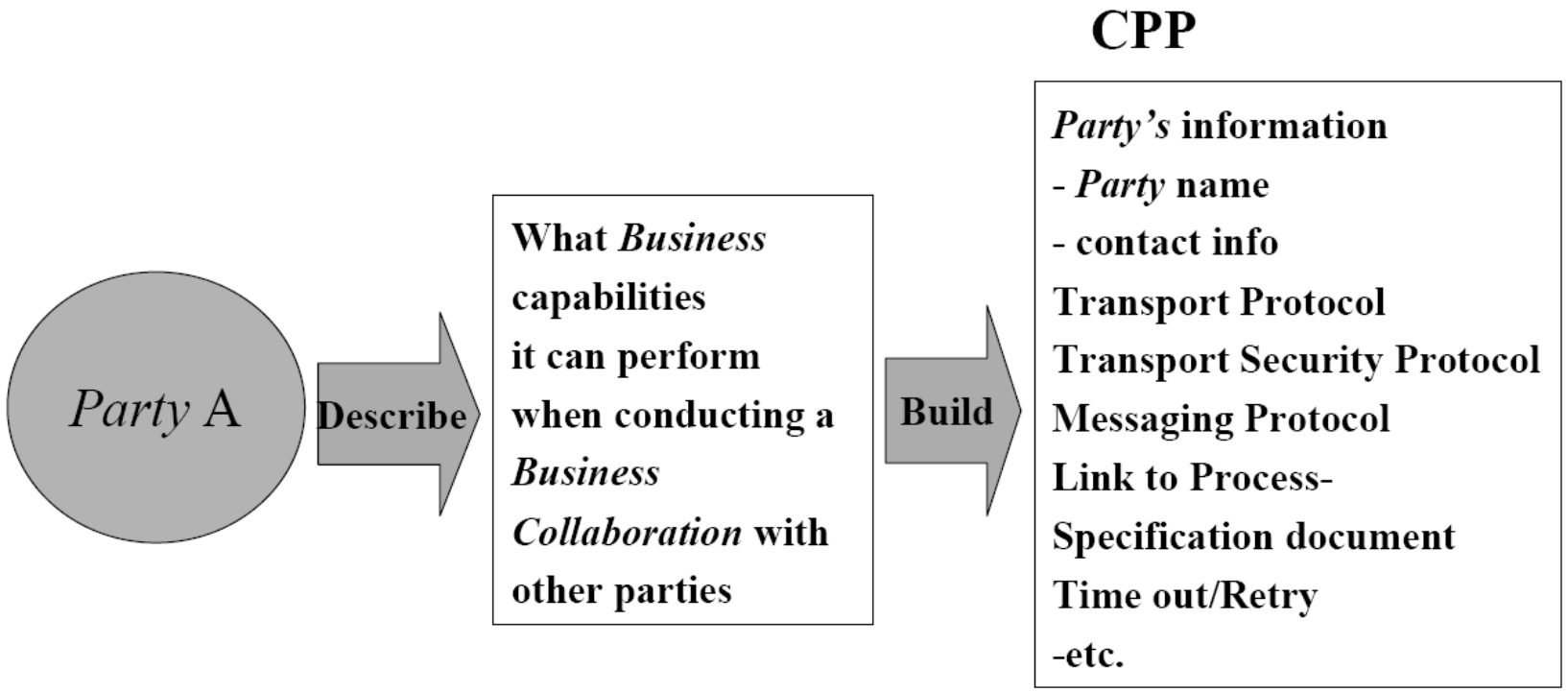


Diagramma UML di una collaborazione binaria

```
<BinaryCollaboration name="Product Fulfillment" timeToPerform="P5D">
   <Documentation>
      timeToPerform = Period: 5 days from start of transaction
   </Documentation>
   <InitiatingRole name="buyer"/>
   <RespondingRole name="seller"/>
   <BusinessTransactionActivity
      name="Create Order"
      businessTransaction="Create Order"
      fromAuthorizedRole="buyer"
      toAuthorizedRole="seller"
      isLegallyBinding=”true”/>
   <BusinessTransactionActivity
      name="Notify shipment"
      businessTransaction="Notify of advance shipment"
      fromAuthorizedRole="buyer"
      toAuthorizedRole="seller"/>
</BinaryCollaboration>
```

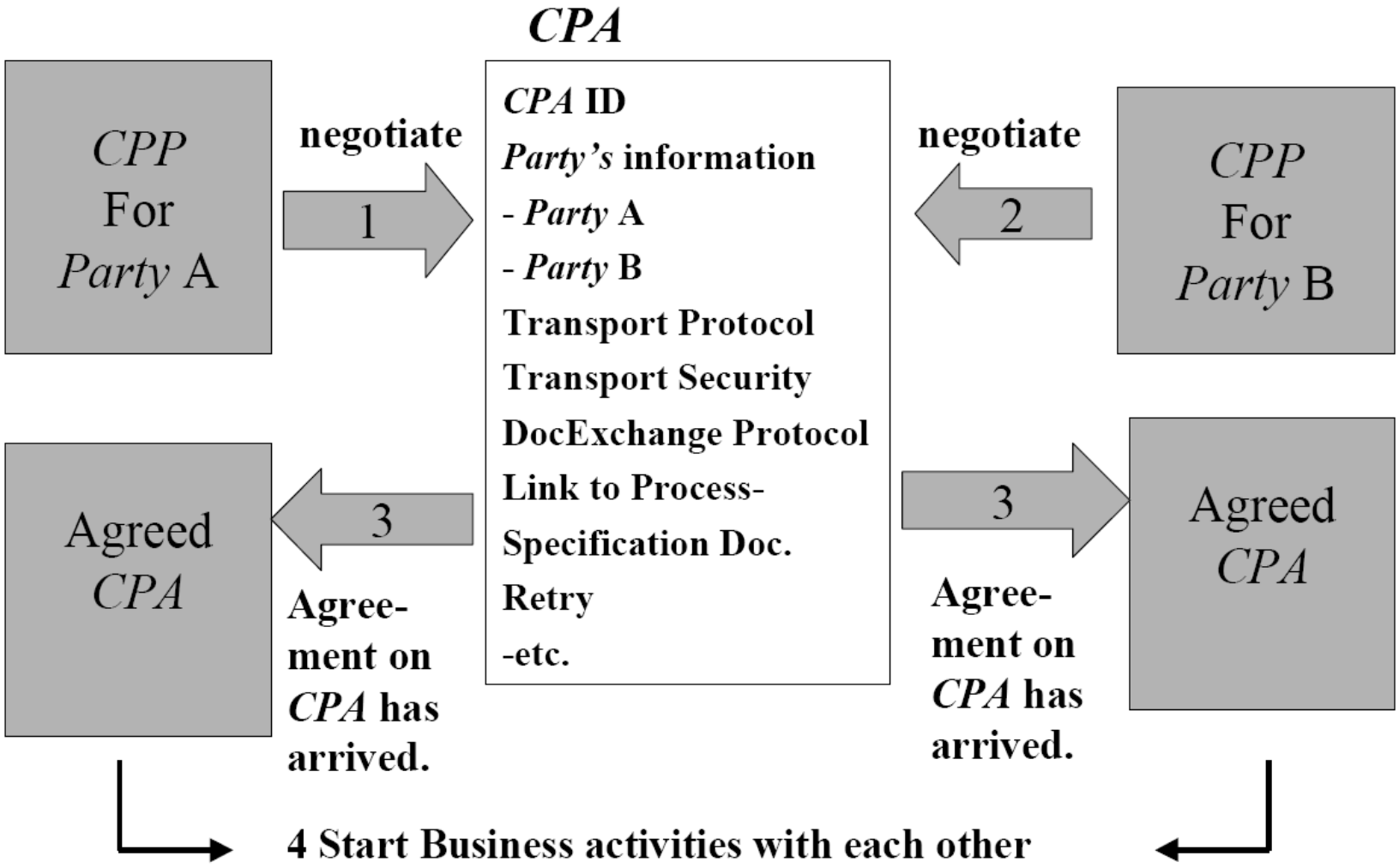
- ✓ *Esempio di Collaboration Protocol Profile*, i CPP possono essere immagazzinati in un apposito repository pubblico quale l’ebXML Registry. Tramite un processo di ricerca ed individuazione, codificato nelle specifiche del repository, una parte commerciale può trovare adeguati Business Partner.



CPP

- ✓ La parte A classifica le informazioni da inserire nel registry per il processo di discovery, costruisce un CPP con queste informazioni e le inserisce in un ebXML Registry.
- ✓ Un CPP è composto da quattro livelli. *Process Specification Layer* (le business transactions che le due parti possono chiedersi e le regole di transizione che determinano l'ordine delle richieste), *Delivery Channels* (le caratteristiche dei messaggi di ricezione e trasmissione), *Document-Exchange Layer* (specifica il

trattamento dei documenti di business es. encryption, digital signature, messaggistica affidabile), *Transport Layer* (protocollo di trasporto per mandare i messaggi nella rete e definire gli indirizzi degli endpoint)



CPA

- ✓ Le due parti usano i loro CPP per costuire congiuntamente un unico CPA calcolando l'intersezione tra i rispettivi CPP. Il risultato definisce come essi possono comportarsi nell'effettuare le loro Business Collaboration.
- ✓ Struttura di un CPP.

```
<tp:CollaborationProtocolProfile
   xmlns:tp="http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd
                       http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd"
   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   tp:cppid="uri:companyA-cpp"
   tp:version="2_0b">
  <tp:PartyInfo>            ... </tp:PartyInfo>    <!-- one or more -->
  <tp:SimplePart id="..."> ... </tp:SimplePart>   <!-- one or more -->
  <tp:Packaging  id="..."> ... </tp:Packaging>    <!-- one or more -->
  <tp:Signature>            ... </tp:Signature>    <!-- zero or one -->
  <tp:Comment> text            </tp:Comment>      <!-- zero or more -->
</tp:CollaborationProtocolProfile>
```

- ✓ *tp* (CPP/CPA namespace), *ds* (XML Digital Signature namespace), *xlink* (Xlink namespace, *Xlink* è un vocabolario XML standardizzato che può essere aggiunto a elementi di documenti istanza XML per descrivere collegamenti estesi, di qualsiasi complessità, fra risorse nel Web)
- ✓ Ciascun *PartyInfo* identifica l'organizzazione e contiene informazioni quali l'id (es. EAN/UCC) i ruoli che può assumere, link a documenti aggiuntivi, certificati,

dettagli relativi alla sicurezza, al protocollo di trasporto. *SimplePart* fornisce una lista di elementi costitutivi identificati dal corrispondente tipo MIME (Multi-purpose Internet Mail Extension, metodo per trasferire ogni tipo di file tramite la posta elettronica) *Packaging* indica informazioni sul Message Header ed il payload dei messaggi.*Signature* abilita il CPA ad essere firmato usando le specifiche XMLDSIG (XML Digital SIGnature specification).

- ✓ *CPA* come detto, è composto da due CPP, pertanto molti elementi sono a comune con questo. Ecco la struttura di alto livello.

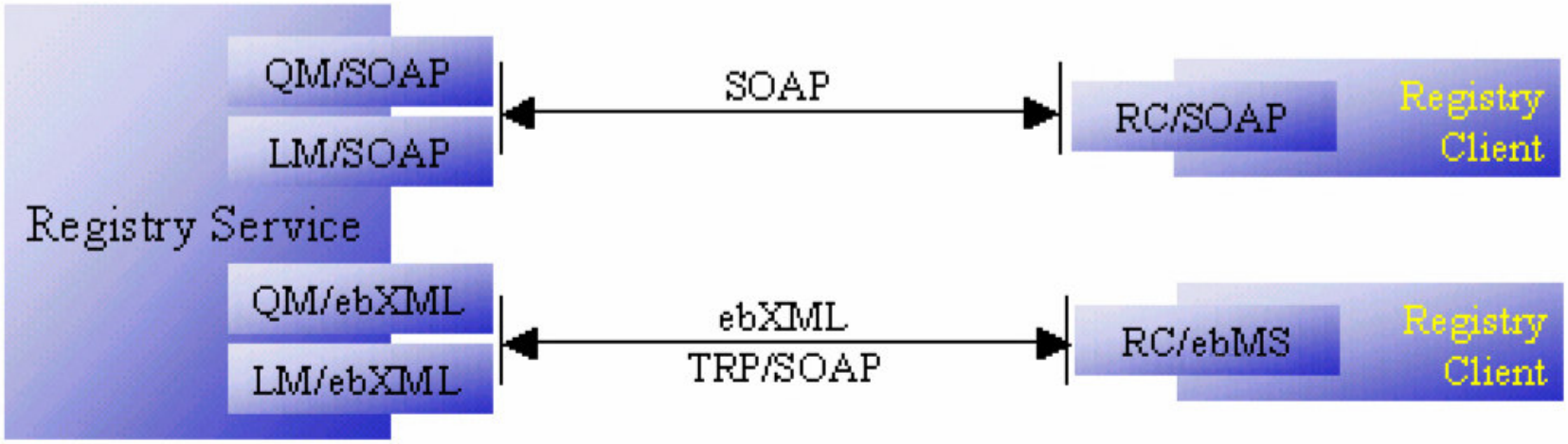
```
<CollaborationProtocolAgreement
   xmlns:tp="http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd"
   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   tp:cpaid="YoursAndMyCPA"
   tp:version="2.0a">
   <tp:Status tp:value="proposed"/>
   <tp:Start>1988-04-07T18:39:09</Start>
   <tp:End>1990-04-07T18:40:00</End>
   <!-- ConversationConstraints MAY appear 0 or 1 time -->
   <tp:ConversationConstraints
      tp:invocationLimit="100"
      tp:concurrentConversations="4"/>
   <tp:PartyInfo> ... </tp:PartyInfo>
   <tp:PartyInfo> ... </tp:PartyInfo>
   <tp:SimplePart tp:id="...">...  </tp:SimplePart>  <!-- one or more -->
   <tp:Packaging  tp:id="...">...  </tp:Packaging>   <!-- one or more -->
   <tp:Signature>              ...  </tp:Signature>   <!-- zero or one time -->
   <tp:Comment xml:lang="en-GB">any text</Comment>   <!-- zero or more -->
</tp:CollaborationProtocolAgreement>
```

- ✓ Dall'esempio viene evidenziata la presenza di due campi PartyInfo che rappresentano le due controparti dell'accordo. Vengono quindi indicati i protocolli di trasporto e di sicurezza scelti e i ruoli assunti dalle due aziende. `Status` indica lo stato del processo che crea il CPA, `Start` ed `End` gli estremi di validità.

EBXML Registry

- ✓ Fornisce un insieme di servizi per la condivisione di informazioni tra le parti: schemi e documenti XML, descrizioni di processi, ebXML Core Component, descrizioni di contesto, modelli UML, informazioni varie circa le parti commerciali e persino componenti software.
- ✓ L'architettura del Registry ebXML consiste di due elementi: il Registry Service (che fornisce i metodi per amministrare un repository) ed i Registry Client (applicazioni che accedono al Registry).
- ✓ Le interfacce primarie del Registry Service sono *Life Cycle Management Interface* (collezione di metodi per gestire gli oggetti nel Registry) e *Query Management Interface*) che controlla la scoperta ed il recupero delle informazioni dal Registry.
- ✓ OASIS/ebXML Registry Information Model (ebRIM) fornisce lo schema ad alto livello per l'ebXML Registry, sia informazioni sui metadati memorizzati nel Registry sia le relazioni fra le classi dei metadati. Il Registry Information Model può

essere implementato in un ebXML Registry sotto forma di uno schema di database relazionale, di uno schema di database ad oggetti, oppure di un altro schema fisico, come un insieme di interfacce e classi all'interno di una Registry Implementation.



✓ Fig. Implementazione concreta in cui il Registry Service supporta collegamenti con diversi protocoli (SOAP ed ebMS)

Core Component

✓ Elementi semantici usati per costruire la struttura semantica di un business document, raggruppati in categorie di *contesto*. Es. la struttura semantica di un *purchase order* avrà elementi semantici quali *item*, *quantity*, *price*, …

✓ Quando il business process context è *Purchasing* ed il geopolitical context è *EU* (European Union) allora il core component *tax.amount* è interpretato come *VAT.amount* (Value Added Tax = IVA).

✓ (Estratto dal Core Component Dictionary)

Category Type	**Basic**
Core Component Type	**amount. type**

CCT (Core Componente Type), Basic, o Aggregate

. . .

Name	tax. amount	***definition***	
UID	000149	The amount of tax.	
Datatype	n/a		
Core Component Type	amount. type		
Core component re-used	n/a		non applicabile
Synonyms		***remarks***	

Naming Convention

object class	tax
property term	amount*
representation type	amount

. . .

Category Type	**CCT**
Core Component Type	**n/a**

Name	amount. type	***definition***
UID	000105	A number of monetary units specified in a currency where the unit of currency is explicit or implied.
UID		
Datatype	n/a	
Core Component Type	n/a	
Core component re-used	n/a	
Synonyms		***remarks***

Naming Convention

object class	amount
property term	type
representation type	

✓ (Altro estratto)

Name	transport. method. code	***definition***
UID	000132	Method of transport used for the conveyance of goods or
UID		persons
Datatype	n/a	
Core Component Type	code. type	
Core component re-used	n/a	
Synonyms	transport mode code	***remarks*** For example, by air, by rail, by sea.
Naming Convention		
object class	transport	
property term	method	
representation type	code	

- ✓ Molti business process sono comuni a tutte le organizzazioni (es. Approvvigionamento, Pagamento, Spedizione).
- ✓ In molti casi i requisiti delle informazioni di business, es. per identificare un prodotto, sono simili.
- ✓ Durante un processo di business, nella medesima comunità di partner commerciali, molti requisiti rimangono inalterati. Ciò che è considerato prodotto, come viene identificato e descritto ecc. rimangono consistenti nella durata del processo.
- ✓ L'obiettivo è quello di fornire un insieme comune di componenti semantici elementari che rappresentano i tipi generali di dati di business correntemente in uso.

- ✓ Inoltre viene definito un modo per creare nuovi vocabolari di business e per ristrutturare quelli già esistenti. Si cerca di garantire rappresentazioni delle informazioni che siano allo stesso tempo leggibili dall'utente e processabili da sistemi automatici.
- ✓ La standardizzazione viene portata avanti secondo un approccio "syntax-neutral", indipendente dalla sintassi. I Core Component sono descritti in UML e conservati nell'ebXML registry. Tale rappresentazione si può convertire in qualsiasi sintassi: XML Schema, XML DTD, …
- ✓ Usare i Core Component come parte fondamentale del framework ebXML aiuta a garantire che due diversi partner commerciali che usano differenti sintassi, utilizzino la semantica di business allo stesso modo, a condizione che le due sintassi siano basate sugli stessi Core Component.

Funzionamento di ebXML

- ✓ Riepilogo dei concetti principali:
  i. meccanismo standard per descrivere processi di business ed i corrispondenti modelli dell'informazione
  ii. meccanismo per registrare e conservare tali entità

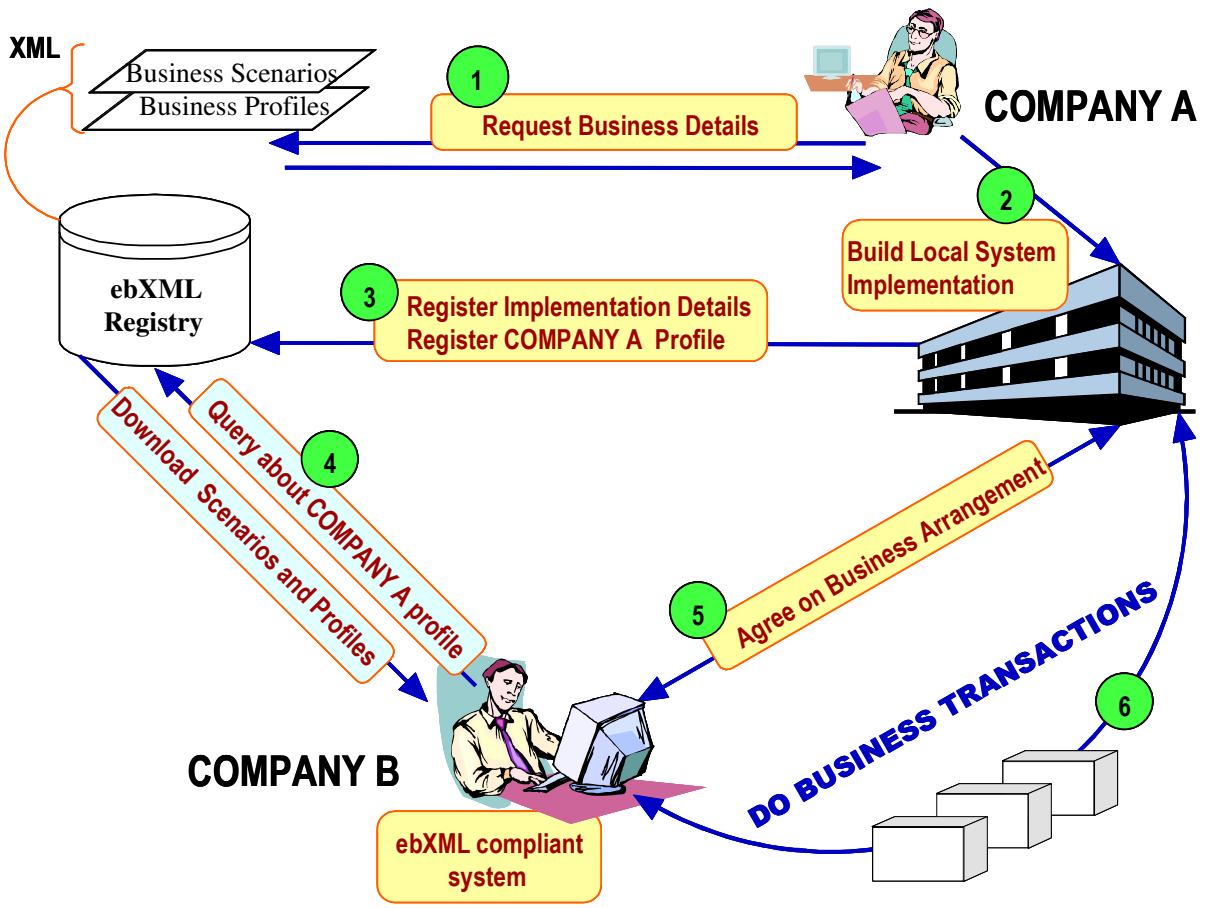
iii. definizione di ciascun partecipante mediante a) i Business Process (BP) che supporta, b) le Business Service Interfaces (BSI) che offre in supporto dei BP, c) i Business Messages scambiati tra le BSI, 4) configurazione tecnica di protocolli di trasporto, sicurezza e codifica.

iv. meccanismo per conservare tali entità

v. meccanismo per descrivere l'esecuzione di disposizioni mutuamente accordate (CPA)

vi. una piattaforma di messaggistica di business standardizzata (ebMS)

vii. meccanismo di configurazione dei servizi di messagistica in accordo con i vincoli definiti nell'accordo di business.



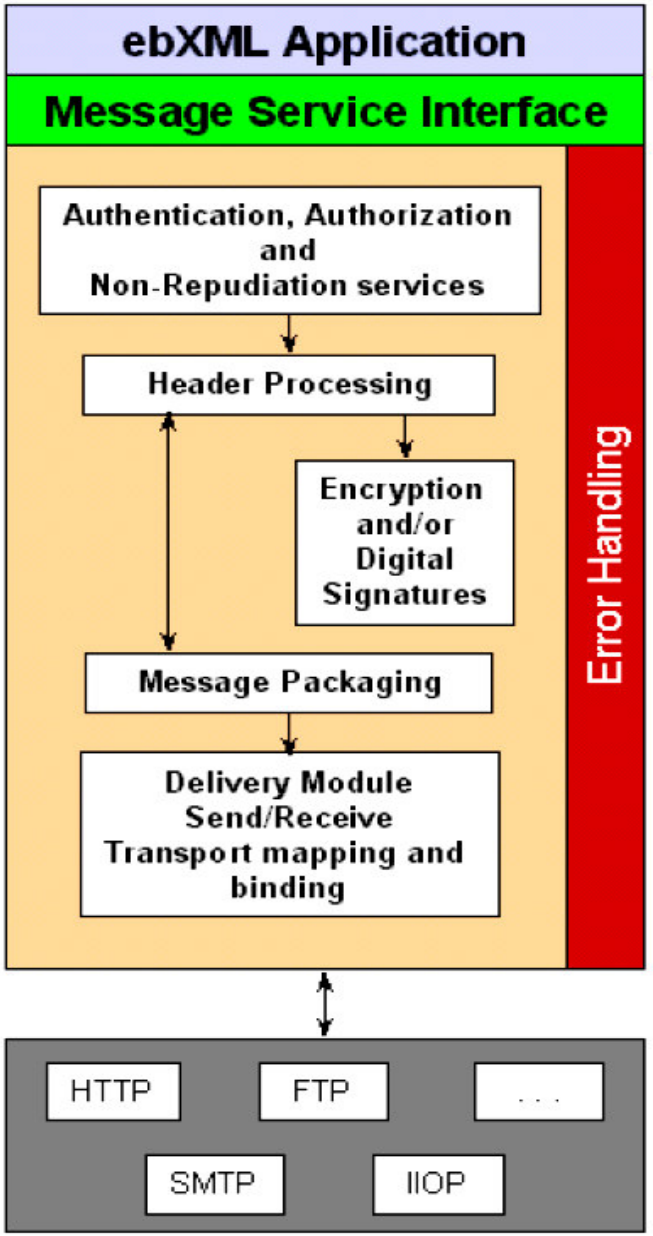
✓ Scenario:

1) La società A viene a conoscenza di un Registro ebXML accessibile via Internet.
2) La società A, analizzato il contenuto del Registry ebXML, decide di realizzare la propria applicazione conforme allo standard ebXML, magari sfruttando applicazioni e componenti ebXML-compliant già disponibili.

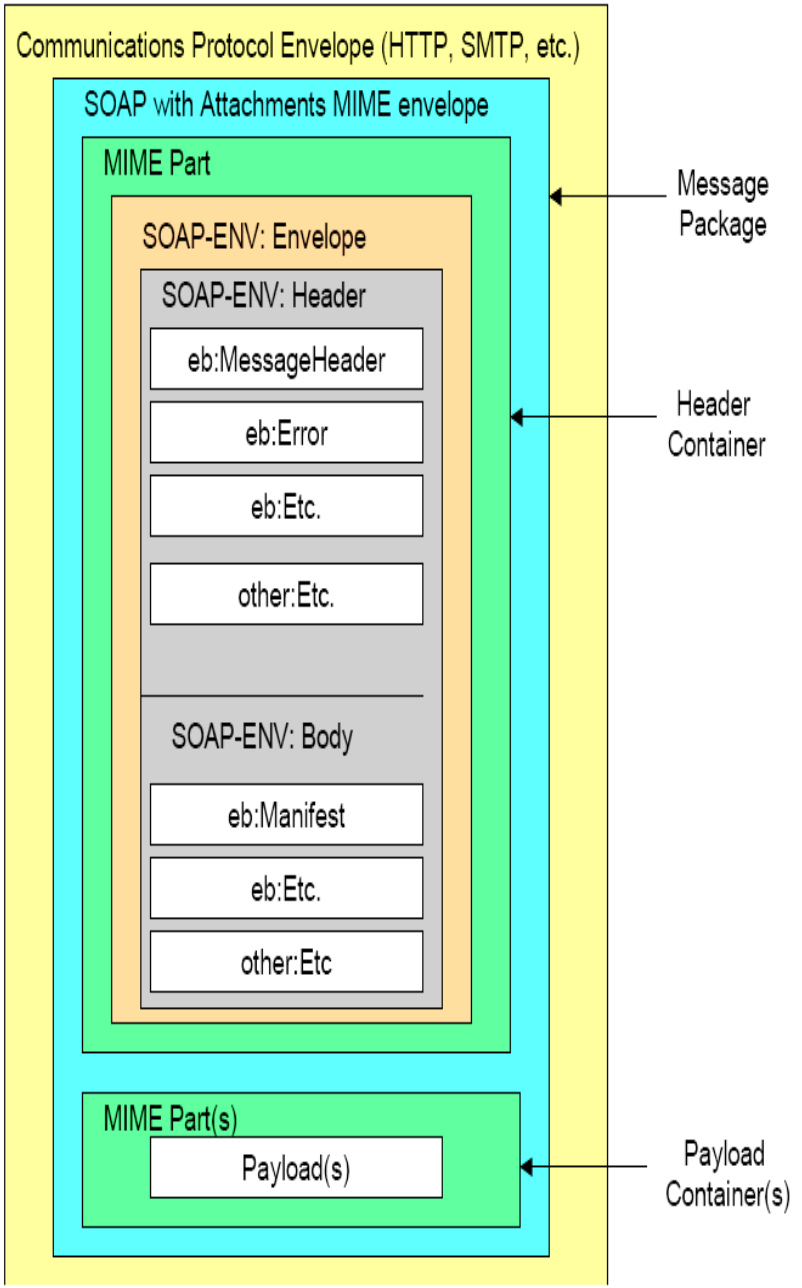
3) La società A invia le informazioni di *Business Profile* al *Registry*. Il Business Profile inviato al Registro descrive capacità e vincoli della società così come tutti gli scenari di business che essa supporta, ossia versione XML dei BP.
4) La società B scopre gli scenari di business supportati dalla Società A nel *Registry*.
5) La società B invia una richiesta alla società A segnalando l'intenzione di impegnarsi in una transazione commerciale appoggiandosi sull'architettura ebXML. La società B si dota di un'applicazione ebXML-compliant ed invia uno schema del business proposto direttamente all'Interfaccia del software ebXML-compliant della società A. Tale schema delinea i processi di business, ed altri specifici requisiti per lo scambio delle informazioni necessarie per il buon fine della transazione, eventuali piani di emergenza ed i requisiti relativi alla sicurezza. La società A accetta la proposta di business.
6) Le società A e B possono ora lanciarsi nell'eBusiness usando ebXML.

ebXML Messaging Service (ebMS)

- ✓ L'ebXML Message Service può essere concettualmente suddiviso in tre parti (i) una Message Service Interface astratta, (ii) funzioni fornite dal Message Service Handler – MSH, (iii) il mapping verso il servizio di trasporto sottostante.
- ✓ *Header Processing*, la creazione dell'ebXML Header Message avviene in base agli input dell'applicazione, passati attraverso la Message Service Interface, e del CPA che decide la struttura del messaggio, e poi informazioni quali firma digitale, identificatori,etc.
- ✓ *Message Packaging*, l'avvolgimento finale di un ebXML message (header+payload) nel messaggio SOAP with attachment container.
- ✓ *Message Service Interface* fornisce delle funzioni :
  - *Send*: spedisce un messaggio ebXML, i valori per i parametri sono ricavati dal Message Header.
  - *Receive*: ricezione dei messaggi
  - *Notify*: fornisce notifica di eventi attesi e inattesi.
  - *Inquire*: fornisce un metodo per conoscere lo stato di un particolare scambio di messaggi.



- ✓ SOAP with Attachment (SwA) è una combinazione del protocollo SOAP con il formato MIME che consente di includere in un messaggio SOAP qualsiasi tipo di dati, in analogia al modello usato per email attachment.
- ✓ L'intero messaggio si compone di tante parti MIME (separate da un opportuno MIME header), la prima delle quali è il SOAP envelope, seguita dagli attachment, ossia i contenuti di livello applicativo.
- ✓ Il SOAP Envelope contiene un Header ed un Body, i quali sono similmente composti da una serie di SOAP block. Il contenuto di questi ultimi viene stabilito dal protocollo di livello superiore, ebXML nella fattispecie.
- ✓ ebXML stabilisce che il SOAP header abbia un elemento `MessageHeader` con informazioni di routing (To/From, etc.) e di contesto del messaggio, ed il SOAP body abbia un elemento opzionale `Manifest` con riferimenti agli allegati ed a documenti esterni.



```
<SOAP:Envelope
  xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://schemas.xmlsoap.org/soap/envelope/
                      http://www.oasis-open.org/committees/ebxml-msg/schema/envelope.xsd">
  <SOAP:Header
    xmlns:eb="http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd"
    xsi:schemaLocation="http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd
                        http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd">
    <eb:MessageHeader ...>    ...   </eb:MessageHeader>
  </SOAP:Header>
  <SOAP:Body
    xmlns:eb="http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd"
    xsi:schemaLocation="http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd
                       http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd">
    <eb:Manifest eb:version="2.0"> ... </eb:Manifest>
  </SOAP:Body>
</SOAP:Envelope>
```

✓ Esempio di MessageHeader e Manifest in uno scenario di creazione nuovo ordine:

```
<eb:MessageHeader SOAP:mustUnderstand="1" eb:version="2.0">
  <eb:From>
    <eb:PartyId>urn:duns:123456789</eb:PartyId>
    <eb:Role>http://rosettanet.org/roles/Buyer</eb:Role>
  </eb:From>
  <eb:To>
    <eb:PartyId>urn:duns:912345678</eb:PartyId>
    <eb:Role>http://rosettanet.org/roles/Seller</eb:Role>
  </eb:To>
  <eb:CPAId>http://esempio.com/cpas/nostrocpaconvoi.xml</eb:CPAId>
  <eb:ConversationId>20001209-133003-28572</eb:ConversationId>
  <eb:Service>urn:services:SupplierOrderProcessing</eb:Service>
  <eb:Action>NewOrder</eb:Action>
  <eb:MessageData>
    <eb:MessageId>20001209-133003-28572@example.com</eb:MessageId>
    <eb:Timestamp>2001-02-15T11:12:12</eb:Timestamp>
  </eb:MessageData>
</eb:MessageHeader>
```

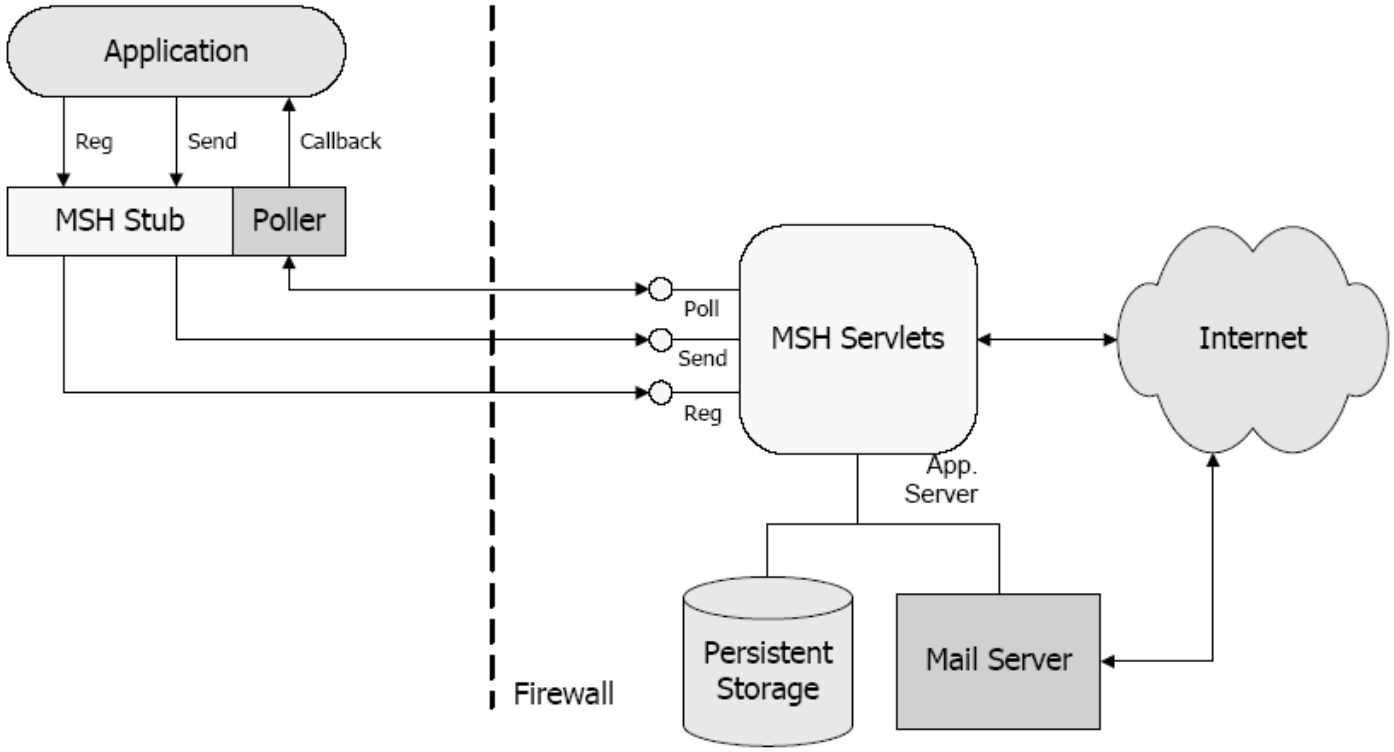
From/To: id del mittente/destinatario e ruolo assunto nella collaborazione.
CPAId: identificatore del CPA, spesso una URI.
ConversationId: identifica l'insieme dei messaggi di una conversazione tra due partner. È inizializzato dalla parte che inizia il dialogo, e verrà mantenuto per tutta la durata degli scambi.
Service: indica il servizio richiesto nel messaggio.
Action: identifica una particolare azione all'interno di un servizio. Ad esempio la creazione di un nuovo ordine.
MessageData: contiene tutte le informazioni necessarie all'identificazione unica di un messaggio, in particolare un id generato casualmente e un timestamp legato all'istante di creazione dell'header. Questo campo può contenere inoltre l'id del messaggio precedente, collegato a quello attuale ed il tempo di vita del messaggio.

```
<SOAP:Body>
  <eb:Manifest eb:version="2.0">
    <eb:Reference xlink:href="cid:ebxmlpayload111@example.com"
                  xlink:role="XLinkRole" xlink:type="simple">
      <eb:Schema eb:location="http://regrep.org/po.xsd" eb:version="2.0"/>
      <eb:Description xml:lang="en-US">Purchase Order 1</eb:Description>
    </eb:Reference>
  </eb:Manifest>
</SOAP:Body>
```

Manifest identifica, attraverso uno o più elementi *Reference*, i documenti allegati al messaggio o risorse esterne accessibili attraverso un URL. Ogni *Reference* a sua volta contiene, nel caso di allegati di tipo XML, un collegamento ad uno schema che ne definisce la struttura ed una descrizione del suo contenuto.

Trasmissione di documenti secondo ebXML

- ✓ *Hermes Message Service Handler* (Hermes MSH) è un sistema di messaggistica ebXML sviluppato dal Center for ECommerce dell' Università di Hong Kong nell'ambito del progetto *Phoenix*, che ha lo scopo di realizzare un'infrastruttura per il commercio elettronico basata sull'utilizzo dello standard ebXML.



- ✓ L'architettura si divide in due parti: una MSH Servlet ed un MSH Stub. Il motivo principale di questa soluzione è legata alla presenza di firewall. Generalmente gli

MSH endpoint sono connessi direttamente ad Internet all'esterno del firewall aziendale. Invece i destinatari dei messaggi ebXML sono applicazioni all'interno del firewall per ovvi motivi di sicurezza.

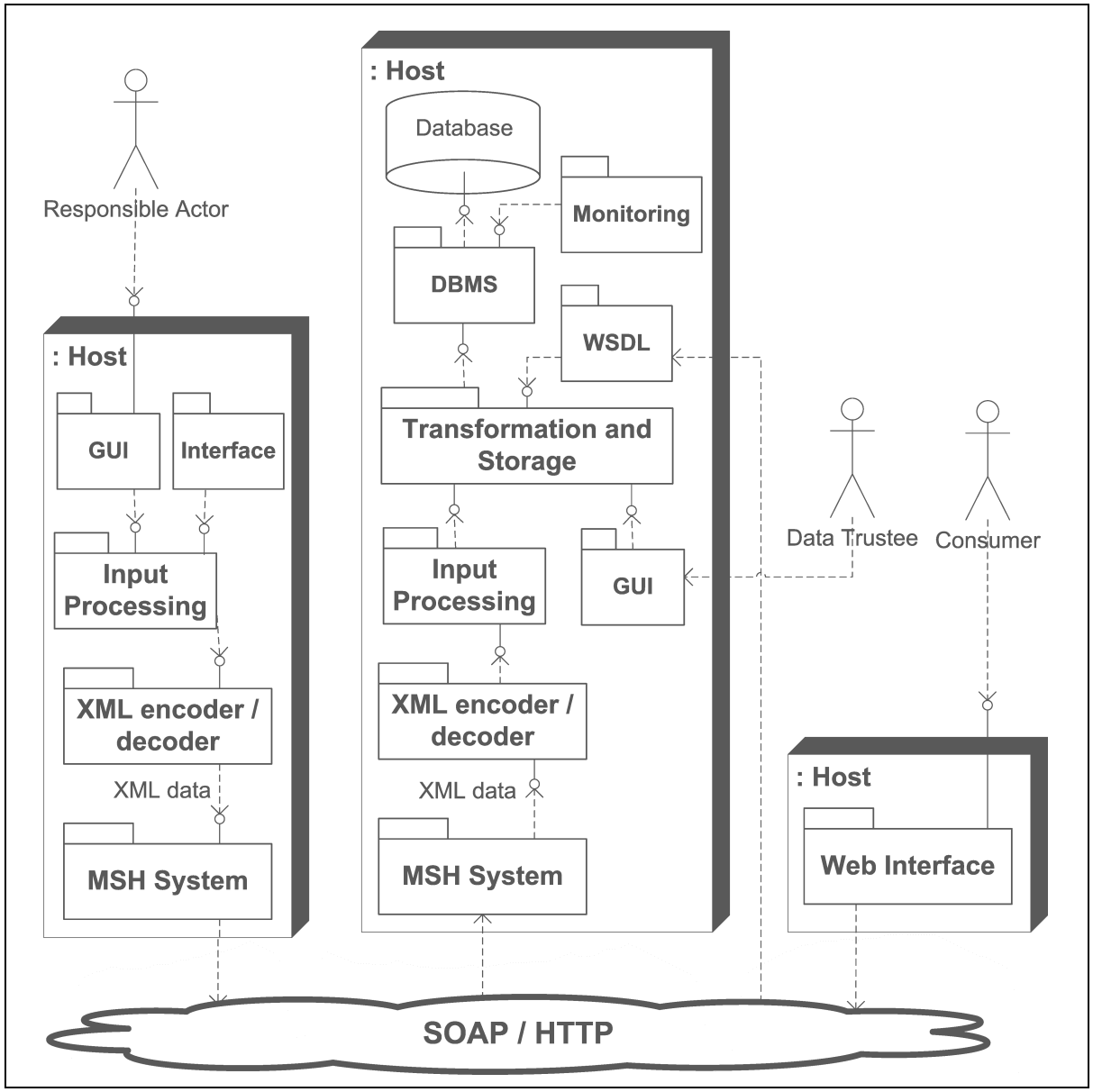
- ✓ Le funzionalità principali di MSH sono quindi implementate nella MSH Servlet, e l'applicazione può gestirle istanziando un MSH Stub e registrandosi mediante una opportuna chiamata di metodo di tale istanza.
- ✓ Le informazioni per la registrazione sono prelevate dal Collaboration Protocol Agreement (CPA) e sono informazioni necessarie per stabilire l'endpoint (indirizzo del partner) a cui inviare il messaggio e per indirizzare i pacchetti in ingresso al giusto destinatario.
- ✓ Una servlet MSH può infatti supportare numerosi client ma ognuno deve registrarsi con un diverso contesto affinché i messaggi siano recapitati correttamente al giusto destinatario.
- ✓ MSH Stub converte i parameri di registrazione in un messaggio SOAP ed invia tale chiamata alla MSH Servlet via HTTP.
- ✓ I messaggi ebXML generati dal client sono inviati allo stesso modo. MSH Stub in tal caso, non solo provvede al packaging del messaggio, ma anche a qualche piccola elaborazione (es. aggiungere firma digitale e criptare) e ad inoltrarlo all'MSH Servlet
- ✓ L'MSH Servlet provvede ad ulteriori elaborazioni (es. la gestione di messaggistica affidabile) ed ad inoltrarlo al destinatario.

- ✓ In ricezione esistono quattro modalità di funzionamento. La più semplice, prevede che MSH Servlet elabori il messaggio e ne individui l'applicazione client di destinazione sulla base delle informazioni di registrazione. MSH stub possiede un polling thread che periodicamente controlla se vi sono messaggi e li invia all'applicazione mediante un metodo callback.
- ✓ Per sviluppare un client ebXML in Java, la classe relativa deve implementare l'interfaccia `MessageListener` e definire i metodi `getClientURL()` (che in modalità base ritorna semplicemente `null`) ed `onMessage()` che viene invocata dal meccanismo di callback quando arriva un messaggio, pertanto contiene tutte le elaborazioni necessarie.
- ✓ La registrazione avviene istanziando un oggetto `Request(…)` in cui viene passato un riferimento all'applicazione (per il meccanismo di callback), il tipo di protocollo scelto, l'end-point dell'applicazione, ed il contesto (composto dal *CPAId*, *ConversationId*, il tipo di servizio richiesto e la particolare azione all'interno di esso)
- ✓ Questi parametri vengono prelevati dal *cpa*, pertanto occorre costruire una classe CPAParser con tutti i metodi per prelevare le informazioni suddette.
- ✓ L'invio avviene istanziando un oggetto EbxmlMessage, che dopo essere stato opportunamento "confezionato" viene spedito nel caso più semplice mediante il metodo `send()`.
- ✓ I messaggi ricevuti da MSH vengono mantenuti in delle apposite cartelle (Repository), mentre gli id degli stessi vengono inseriti in un database mysql insieme

ad altre informazioni prelevate dall'intestazione. Nel file di configurazione msh_client_property.xml è possibile indicare il tempo di attesa, espresso in secondi, fra un polling e il successivo.

- ✓ Le operazioni di ricezione sono invocate all'interno di `onMessage()` e riguardano l'estrazione del payload, la costruzione di un oggetto Document e quindi l'elaborazione del contenuto.

Architettura di cerere semplificata



Free ebXML implementations

http://www.freebxml.org/projects.htm

Webswell3 Broker (WB)

- ✓ Modulo di integrazione tra business application e sistema locale di messaggistica. Può essere usato come modulo di **Webswell Connect**, di **Hermes MSH**, di **freebXML Registry**.

- ✓ Comunica con applicazioni legacy attraverso l'esportazione ed importazione di dati nel file system, e viene configurato mediante opportuni CPA.

freebXML Core Components (CC)

- ✓ Progetto in fase di sviluppo, basato sulla specifica ebXML Core Components, finalizzato alla realizzazione di API Java per facilitare la gestione di vocabolari da

[3] La Webswell è una organizzazione specializzata in soluzioni e consulenze su web service integtration. Relativamente ad ebXML, ha sviluppato il Webswell Broker e Webswell Connect (http://www2.webswell.com/products/ download/), open source e free per le università.

parte di esperti di dominio, usando il **freebXML Registry** e le API **JAXR** (Java API for XML Registries).

open source ebXML Registry

- ✓ Un XML registry/repository che consente un accesso (personalizzabile sulla base del ruolo) ad informazioni critiche per il business mediante protocolli standard.
- ✓ Il repository può archiviare dati XML, web services, or qualsiasi altro tipo di dato, ed il registry aministra l'intero ciclo di vita di tali informazioni mediante una struttura di meta dati.
- ✓ Principali funzioni:
  - funzionalità di Web Content Management
  - controllo degli accessi basatio sui ruoli (politiche XACML)
  - Registry Browser configurabile mediante schemi XML
  - Supporto per modo read-only in caso di accesso non autenticato
  - intergrazione con Web browser

Hermes MSH

- ✓ Sistema di messaggistica adoperato nel prototipo *cerere04*, ora giunto alla release 1.001 il 29 agosto 2005.

ebMail

- ✓ Interfaccia grafica (del tipo client di posta elettronica) che aiuta utenti con conoscenze mnimali di ebXML ad entrare in attività B2B. Utile per piccome medie aziende che non necessitano di integrazione di backend.

Webswell Connect

- ✓ Piattaforma di e-business che consente di esporre i propri profili e cercare partner commerciali secondo vari criteri.

- ✓ Archiviazione e classificazione di artefatti di e-business quali profili di partenr commerciali, definizioni di tipi di documenti, XML schema, Collaboration Protocol Profiles (CPP), Collaboration Protocol Agreements (CPA), Business Process Specification Schemas (BPSS) etc.

Collaboration Protocol Agreement Negotiation Tool[4]

- ✓ Progetto in fase di sviluppo per creare, derivare o negoziare un ebXML Collaboration Protocol Agreement (CPA) a partire da due ebXML Collaboration Protocol Profiles (CPP), oppure da un ebXML CPA template.

[4] Progetto condotto dalla Curtin University of Technology (Australia) http://www.schlegel.li/ebXML/index.html