Web Services

XML, WSDL, SOAP und UDDI
Einblicke und Ausblicke

Alternativen - ebXML

- Inhalt
  - Übersicht über ebXML
    - Ein einfaches Beispiel
  - Die ebXML Spezifikation
    - Architekturübersicht

Alternativen - ebXML

- *Die ebXML Initiative verlief parallel zur Entwicklung der Web Service Konzepte und verfolgte die gleichen Ziele.*
  - Eine Gruppe, von der UN gesponsert und durch die OASIS (Organizaton for the Advancement of Structured Information Standards) startete die ersten ebXML Aktivitäten.
  - Ziel
    - Globale Standards für den elektronischen Handel schaffen.
    - Effizienzsteigerung und
    - Kostenreduktion

Alternativen - ebXML

- *Die ebXML Initiative nimmt an, dass in absehbarer Zeit Geschäftsdaten elektronisch über das Web ausgetauscht werden können.*
  - Bestellungen
  - Rechnungen
  - Produktinformationen

    jeweils als XML Dokumente.

  - ebXML legte sich auf SOAP Messages mit Attachments als Standard-Austausch fest
    - Damit konvergierte ebXML mit SOAP, UDDI, WSDL
    - Zusätzlich verlangt ebXML höhere Sicherheit und Transaktionen.

Übersicht über ebXML

- *ebXML definiert die Interaktion von Geschäftsprozessen.*
  - SOAP, WSDL und UDDI sind der Versuch RPC für den XML Dokumentaustausch aufs Internet zu bringen.
  - ebXML hat vom Start weg das Ziel Business-Informationen mittels XML zu übermitteln und auszutauschen, analog zu EDI / EDIFACT, mit den

    Zielen:
  - Kostenreduktion (generell und im Vergleich zu EDI)
  - Effizienzsteigerung (mittels vordefinierter Prozesse)

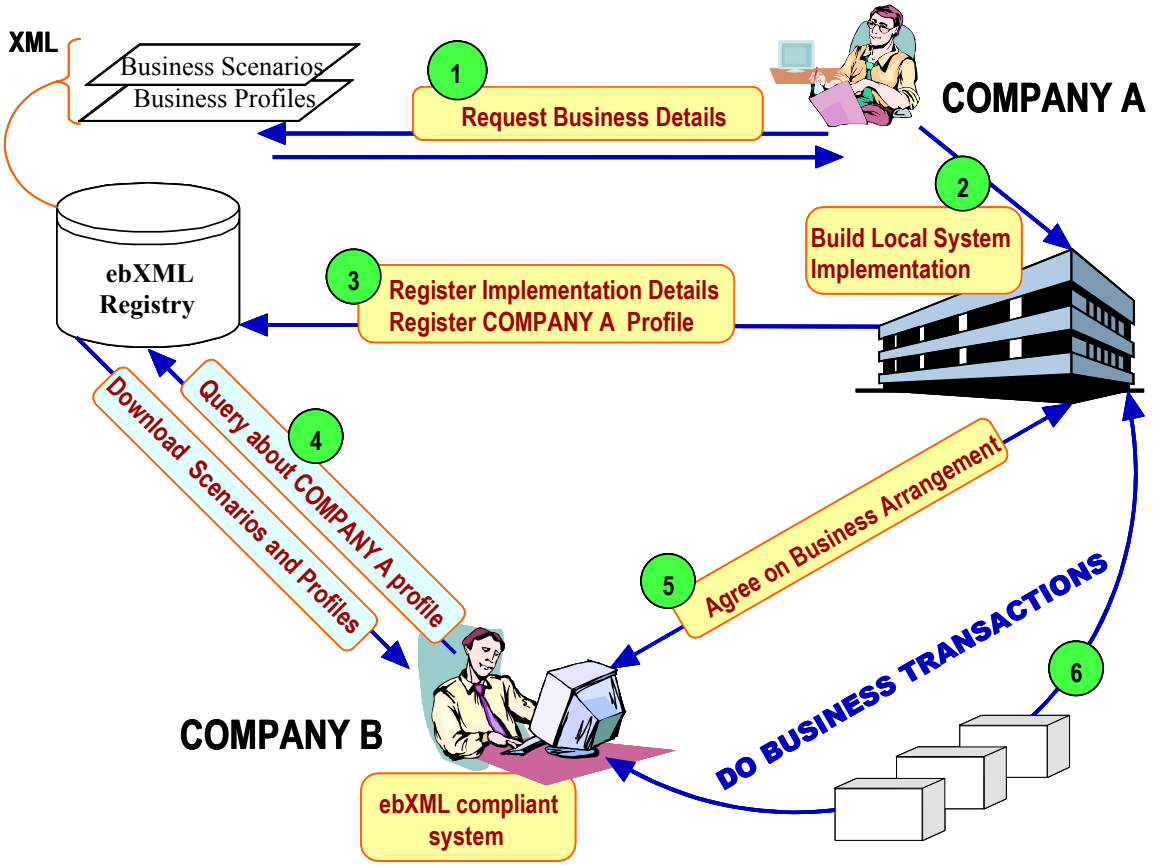
    im e-Commerce

Übersicht über ebXML – ein Beispiel

- *Als erstes muss der Geschäftsfall identifiziert werden.*
  - Zuerst wird der Geschäftsfall festgelegt.
  - Daraus werden Informationsbedürfnisse hergeleitet.
  - Beispiel (einzelne Schritte können optional sein):
    - Whoopy sendet einen Rahmenkaufvertrag an Goopy, den Hersteller.
    - Goopy bestätigt diesen Rahmenvertrag (in etwa ein Budget).
    - Whoopy fragt gelegentlich nach, ob die bestellte Ware geliefert werden kann und wann.
    - Goopy sendet die Ware und einen Lieferschein.
    - Whoopy sendet Goopy den Warenempfangsschein.
    - Goopy sendet Whoopy die Rechnung.
    - Whoopy überweist den Betrag.
    - Goopy sendet eine Bestätigung des Zahlungseingangs.

Übersicht über ebXML – ein Beispiel

- *ebXML im Überblick.*
  - *Quelle:* ebXML Specification [ebTA.doc]



Übersicht über ebXML – ebXML und EDI

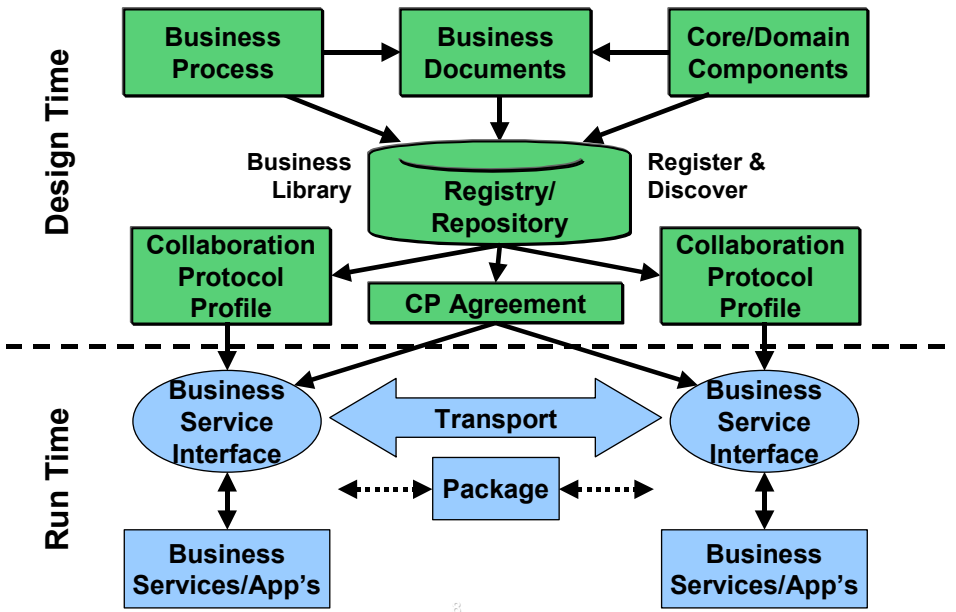
- *ebXML wird als mögliche Ablösung von EDI gesehen.*
  - ebXML ist kostengünstiger, der Unterhalt ist einfacher als bei EDI
  - XML als Basis hat einiges zu bieten:
    - Es ist einfacher als EDI
    - Es kann für viele weitere Aufgaben als nur den Datenaustausch eingesetzt werden.
    - Entwickler mit XML Kenntnissen lassen sich leichter finden als EDI Experten.
    - XML ist eine Plattform-neutrale Sprache

Übersicht über ebXML – ebXML bietet mehr

- *ebXML definiert eine Architektur für Geschäftsabläufe.*
  - ebXML definiert nicht nur eine Grammtik (wie MathML, Scalable vector Graphics SVG, u.a.).
  - ebXML definiert eine Architektur mit geänderten Geschäftsabläufe und dokumentiert diese.
  - In ebXML kann definiert werden, wie Firmen ihre Geschäftsabläufe strukturieren (Business Process Specifications).
  - Aus vordefinierten Kern-Komponenten werden Dokumente zusammengestellt.
  - Messages verwenden Standard-Formate und –Protokolle.
  - Informationen über diese Abläufe, Dokumente und Geschäft-Profile werden in ebXML Registries gespeichert, müssen also nicht immer neu erfunden werden.

Übersicht über ebXML – ein Beispiel

- *ebXML in der Übersicht.*



Übersicht über ebXML – ebXML bietet mehr

- *Wie findet eine Firma Geschäftspartner?*
  - Registries enthalten auch wichtige Informationen zum Thema „Trading Partner“.
  - In ebXML werden Geschäfte als Austausch von Dokumenten beschrieben (Bestellungen, Lieferscheine, u.a.m.)
  - Die Registry speichert Informationen über potentielle Trading Partner in der Form von *Collaboration Protocol Profiles* (CPPs)
    - CPPs sind XML Dokumente, welche mithilfe eines speziellen Vokabulars de Geschäftsprozesse identifiziert, welche eine Firma anbietet und an denen sie teilnehmen kann.
    - Zudem enthält die Beschreibung technische Informationen.
    - Beispiel:
      - Die Firma akzeptiert Online Bestellungen über HTTPS

Übersicht über ebXML – ebXML bietet mehr

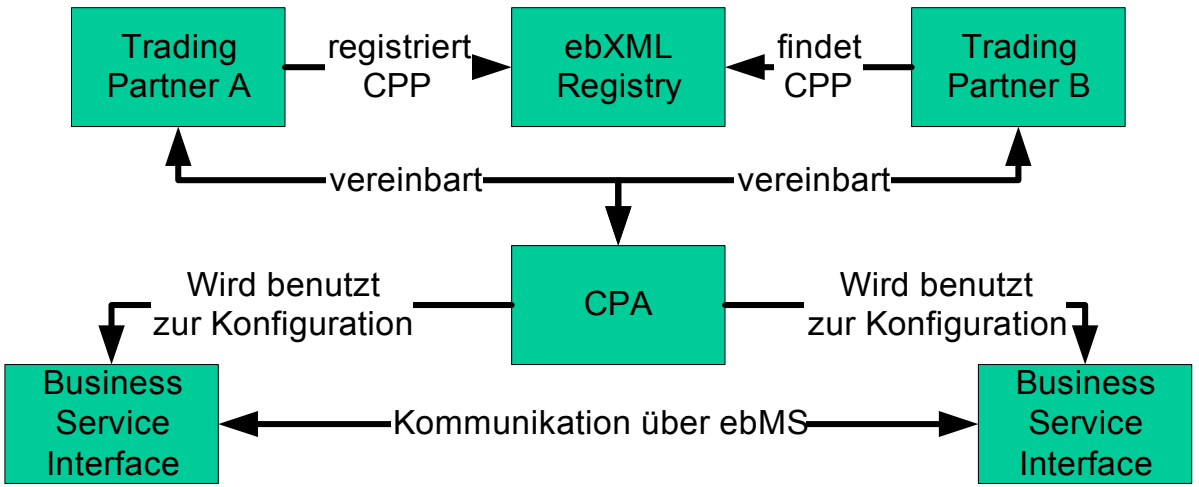
- *Geschäfte knüpfen*
  - Nachdem wir den Handelpartner gefunden haben, müssen die Systeme so konfiguriert werden, dass sie zusammenarbeiten, Transaktionen ausführen können.
  - Dies geschieht mithilfe des *Collaboration Protocol Agreement* (CPA).
    - CPAs bestehen aus CPPs (Collaboration Protocol Profile) für jeden Handelspartner. Diese legen fest, welche Kollaborationen möglich sind und wie.
    - Die CPPs können auch technische Informationen enthalten, wie etwa Hinweise auf Protokolle und Anforderungen betreffend Verifikation und Bestätigungen.
  - Mit dem CPA kann auf beiden Seiten die Applikation, das *Business Service Interface* (BSI) konfiguriert werden.

Übersicht über ebXML – ebXML bietet mehr

- *Was bietet ebXML sonst noch? Fortgeschrittene Features.*
  - Legale Aspekte
    - ebXML bietet die Möglichkeit legal verbindliche Transaktionen zu spezifizieren.
  - Sicherheit
    - ebXML enthält einen technischen Report *Technical Architecture Risk Assessment v 1.0* in dem die relevanten Themen behandelt werden.

ebXML Architektur

- *Systemübersicht.*



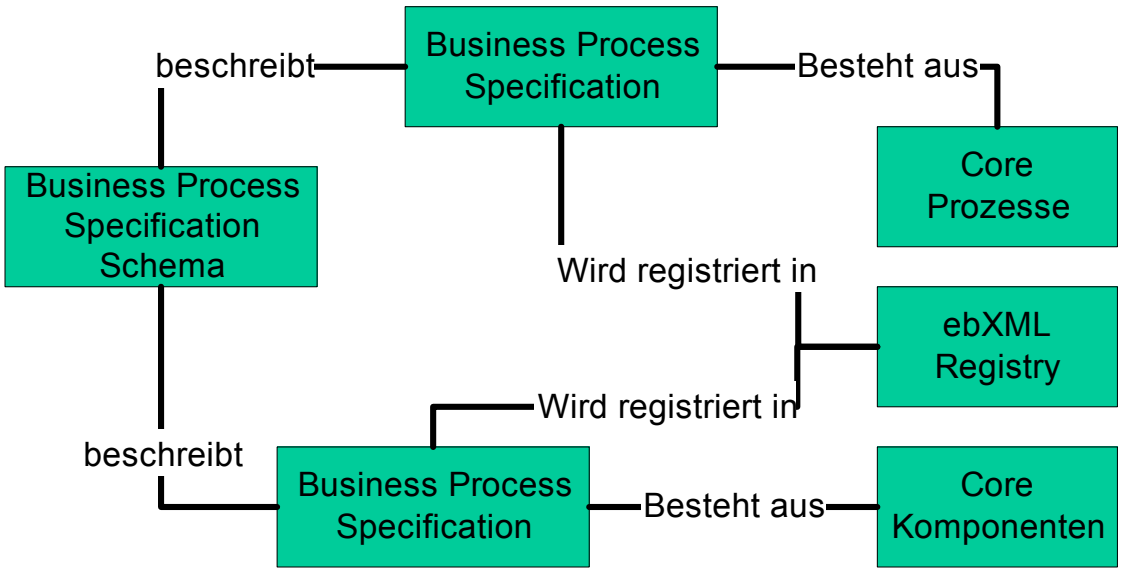
- *CPP = Collaboration Protocol Profile*
- *CPA = Collaboration Protocol Agreement*

ebXML - Architektur

- *Systemübersicht*
  - Nachdem Sie eine Grundidee haben, wie ebXML funktionieren könnte, kümmern wir uns um die Architektur.
  - Diese umfasst einige zum Teil überlappende Technologien.
  - Schauen wir uns als erstes den Prozess als Ganzes an. Oberflächlich gesehen ist ebXML recht einfach
    - Zuerst wird ein Handelspartner gesucht.
    - Dann wird ein CPA (Collaboration Protocol Agreement) kreiert.
      - Beide Handelspartner müssen sich über das CPA einigen.
    - Mithilfe des CPA werden die Business System Interfaces konfiguriert.
    - Und schliesslich startet der Geschäftsablauf.

ebXML - Architektur

- *Business Prozesse und Business Dokumente*

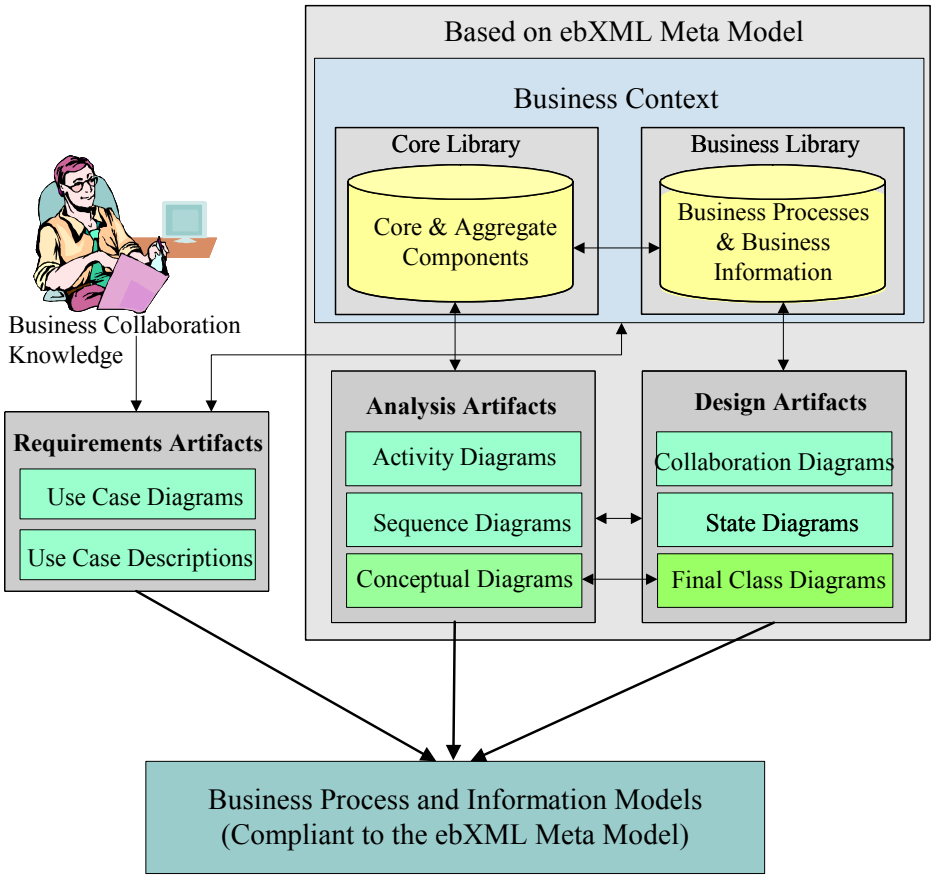


ebXML - Architektur

- *Business Prozesse und Business Dokumente*
  - Business Prozesse und Business Dokumente werden vor ihrem Einsatz entworfen und dokumentiert.
  - Normalerweise werden sie aus existierenden Komponenten und Prozessen zusammengestellt.
  - Beispiel:
    - Business Prozesse können aus Core Prozessen zusammengestellt werden.
    - Diese sind in einer Business Library registriert.
  - Business Dokumente werden normalerweise aus existierenden Core Komponenten aus einer Registry zusammengestellt.
  - Prozesse und Dokumente werden im *Business Process Specification Schema* dokumentiert und in einer ebXML Registry gespeichert.
    - Sie stehen damit für CPPs und CPAs zur Verfügung.

ebXML - Architektur

- *Unified Modeling Methodology*



ebXML - Architektur

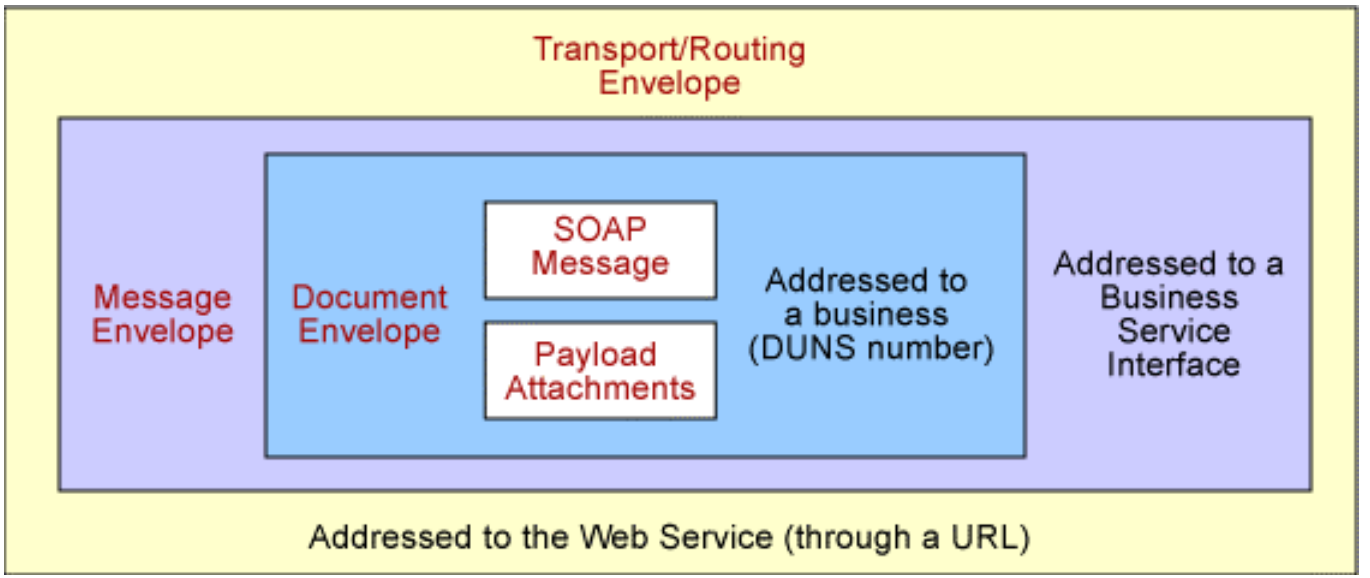
- *Business Process Specification Schema*
  - Business Modelle definieren, wie die Business-Prozesse gefunden, definiert und dokumentiert werden.
  - In ebXML kann dies in UMM, der *Unified Modeling Methodology* geschehen.
    - UMM ist aber nicht zwingend vorgeschrieben, es ist aber eine gemeinsame Sprache.
  - Die BPSS ist ein Subset von UMM.
    - Typischerweise beschreibt man BPSS in UML (UMM basiert auf RUP und UML) und übersetzt die Beschreibung in XML Schema oder DTD mithilfe von Produktionsregeln.

ebXML - Architektur

- *Das Registry Informations-Modell*
  - Nach der Definition der Business Prozesse und Dokumente werden diese in der ebXML gespeichert, zusammen mit CPPs und CPAs, Klassifikationen und anderen Objekten.
  - Die Organisation dieser Informationen geschieht mithilfe des *Registry Information Model*.
  - ebXML Registries speichern nur die Meadaten der aktuellen Dokumente.
  - ebXML Registries bestehen aus ***`RegistryEntry`*** Objekten unterschiedlichen Typs:
    - **`Organization`**
    - **`Package`**
    - **`Slot`** (Informationen über Objekte)
    - **`Association`** (zwischen Objekten)
  - Die Registry kann abgefragt oder Einträge mutiert werden.

ebXML - Architektur

- *Message Struktur*
  - ebXML verwendet ein offenes Message Format, damit Erweiterungen später jederzeit möglich sind.
  - Der *ebXML Service* (**ebMS**) basiert auf SOAP mit Attachments.



ebXML - Business Prozesse

- *Analyse des Geschäftes*
  - Business Prozess müssen standardisiert beschrieben werden. Dies geschieht in der *Business Process Specification,* sinnvollerweise in UMM / UML.

  - Ziel der Analyse ist die Definition der Business Prozesse und Business Informationen:
    - *Business Prozesse* beschreiben, was die Firma wie macht
    - *Business Informationen* beschreiben das Geschäft, sind die Informationen, mit deren Hilfe Geschäfte abgewickelt werden.
    - *Business Dokumente* halten die Business Informationen fest.

ebXML - Business Prozesse

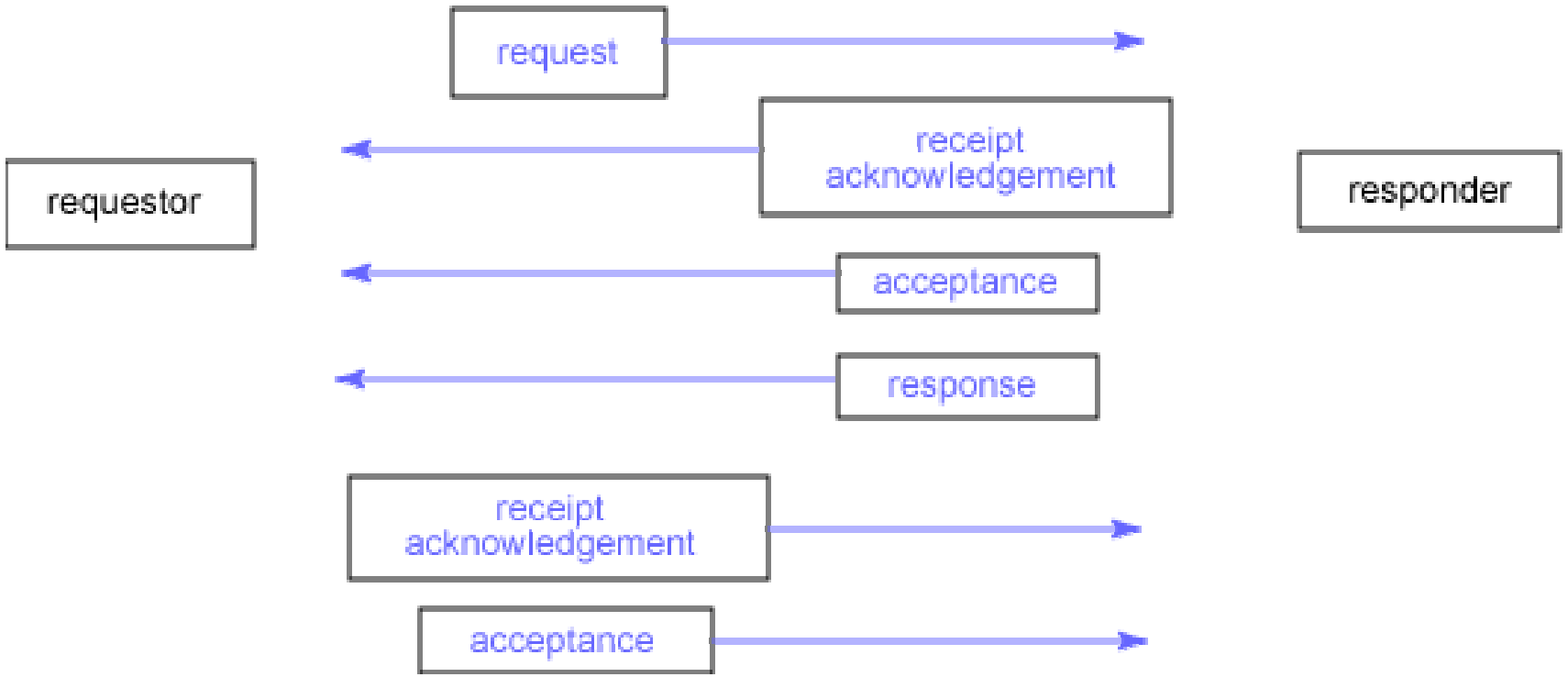
- *Business Collaboration*
  - Eine Business Collaboration orchestriert ein Set von *Business Transaction Activities*, in dem zwei Handelspartner Dokumente austauschen.
  - ebXML unterscheidet zwei Typen von Collaborations:
    - *Binary Collaborations* für den Austausch von Dokumenten und
    - *Multipart Collaboration*, falls Informationen zwischen mehreren Handelspartnern ausgetauscht wird.
      Diese setzen sich aber letztlich aus orchestrierten Binary Collaborations zusammen.
  - Auf der untersten Ebene bestehen Collaborations aus Transaktionen.
  - Collaborations können auch verschachtelt sein:
    - Eine Business Transaktion kann auch eine Collaboration sein.

ebXML - Business Prozesse

- *Business Transaktionen*
  - Die Business Transaktion ist in der Regel die atomare Arbeitsstufe eines Business Prozesses:
    - Sie kann entweder erfolgreich abgeschlossen werden oder wird rückgängig gemacht.
  - In einer Business Transaktion tauschen Handelspartner Dokumente aus.
    - Der Partner, der das Dokument anfordert, wird Requestor genannt.
    - Der andere Partner ist der Responder.
      - Der Responder kann entweder
      - ein Dokument,
      - eine Bestätigung oder
      - keine Antwort senden.

ebXML - Business Prozesse

- *Transaktionsablauf*
  - Business Transaktionen und deren Überwachung



ebXML - Business Prozesse

- *Business Transaktionen und deren Überwachung*
  - Der Requestor sendet eine Anfrage
    - Die Kontrolle der Transaktion wird an den Responder übergeben.
  - Der Responder liest die Nachricht und
    - Sendet eine Empfangs-Bestätigung an den Requestor
    - Dann eine Akzeptanz-Bestätigung und
    - Schliesslich die eigentliche Antwort.
    - Damit wird die Kontrolle der Transaktion wieder an den Requestor abgegeben.
  - Der Requestor seinerseits sendet nun
    - Eine Empfangs-Bestätigung an den Responder
    - Dann eine Akzeptanz-Bestätigung.

ebXML - Business Prozesse

- *Choreografie und Zustände*
  - Eine Collaboration beseht aus orchestrierten Busines Transaktionen.
  - Die Choreografie wird als Folge von Zuständen und Übergängen beschrieben.
  - Eine *Business Activity* beschreibt einen abstrakten Zustand, die Business Transaktions-Aktivitäten beschreiben den konkreten Zustand.
    - **`start`**, **`fork`**, **`synchronization`** und **`completion`** (**`success`** oder **`failure`**) sind Hilfszustände.
    - Transaktionen werden durch Wächter (*guards*) überwacht, Sie haben die Funktion eines *gates*, mit dem kontrolliert wird, ob die Transaktion stattfindet oder nicht.
  - Die gesamte Logik des Zustandsmanagements ist Teil des Business Service Interfaces der Applikation.

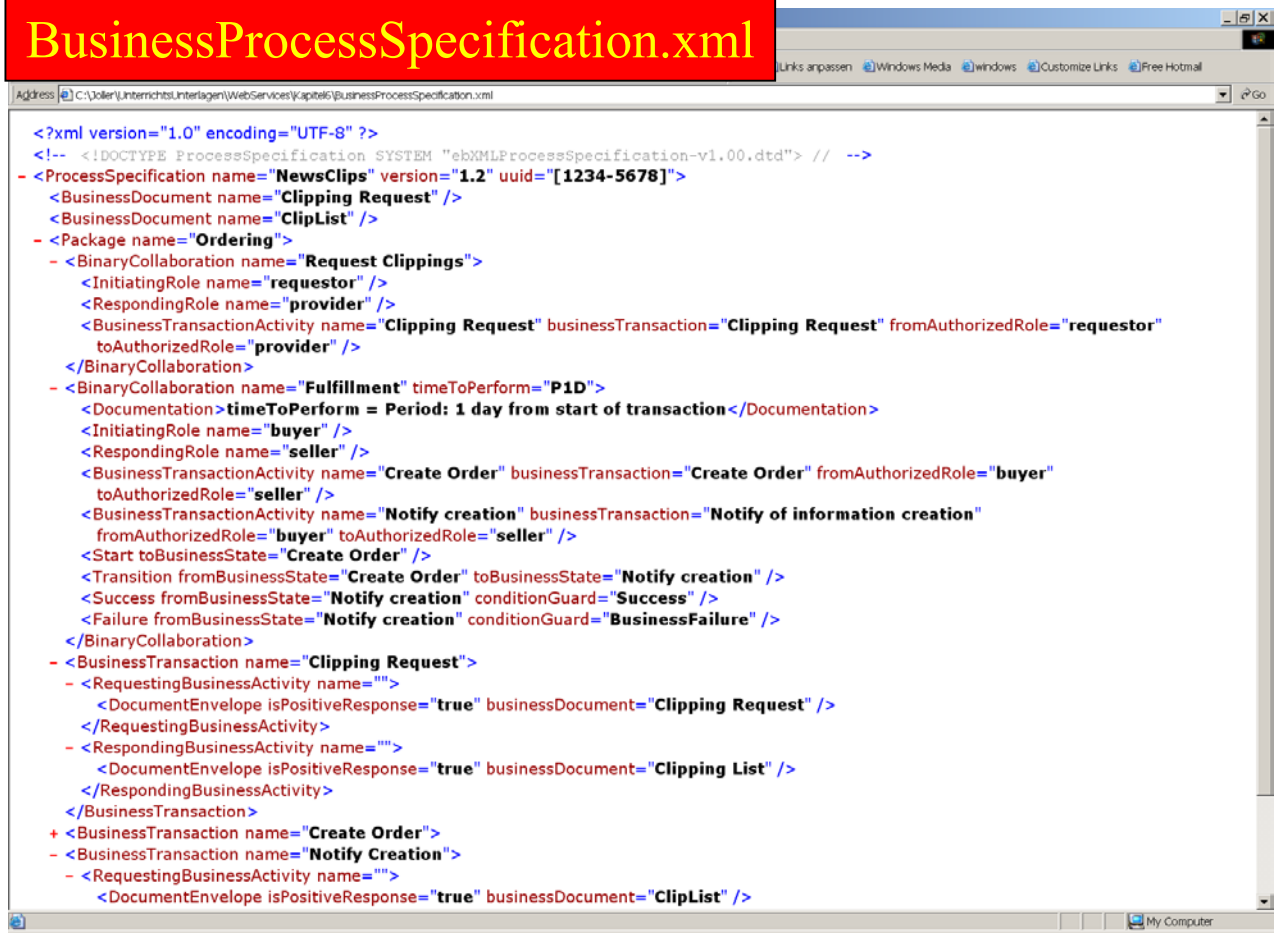
ebXML - Business Prozesse

- *Business Dokumente – woher stammen diese?*
  - Business Dokumente bestehen aus *Business Information Objects*.
    - Diese bestehen nicht aus Informationen, sondern aus XML Schemas und DTD‘s mit deren Hilfe die Informationsstruktur festgehalten wird, damit der Empfänger sie korrekt interpretieren kann.
  - ebXML versucht *Core Components* zu definieren, welche in der der *Core Library* gespeichert werden und für die Definition von Business Dokumenten eingesetzt werden können.
    - Die Core Library ist noch nicht vollständig definiert, wird dies eventuell auch nie ganz sein.

ebXML - Business Prozesse

- *Beispiel für eine Business Process Specification*

BusinessProcessSpecification.xml



```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- <!DOCTYPE ProcessSpecification SYSTEM "ebXMLProcessSpecification-v1.00.dtd"> // -->
- <ProcessSpecification name="NewsClips" version="1.2" uuid="[1234-5678]">
    <BusinessDocument name="Clipping Request" />
    <BusinessDocument name="ClipList" />
  - <Package name="Ordering">
    - <BinaryCollaboration name="Request Clippings">
        <InitiatingRole name="requestor" />
        <RespondingRole name="provider" />
        <BusinessTransactionActivity name="Clipping Request" businessTransaction="Clipping Request" fromAuthorizedRole="requestor"
          toAuthorizedRole="provider" />
      </BinaryCollaboration>
    - <BinaryCollaboration name="Fulfillment" timeToPerform="P1D">
        <Documentation>timeToPerform = Period: 1 day from start of transaction</Documentation>
        <InitiatingRole name="buyer" />
        <RespondingRole name="seller" />
        <BusinessTransactionActivity name="Create Order" businessTransaction="Create Order" fromAuthorizedRole="buyer"
          toAuthorizedRole="seller" />
        <BusinessTransactionActivity name="Notify creation" businessTransaction="Notify of information creation"
          fromAuthorizedRole="buyer" toAuthorizedRole="seller" />
        <Start toBusinessState="Create Order" />
        <Transition fromBusinessState="Create Order" toBusinessState="Notify creation" />
        <Success fromBusinessState="Notify creation" conditionGuard="Success" />
        <Failure fromBusinessState="Notify creation" conditionGuard="BusinessFailure" />
      </BinaryCollaboration>
    - <BusinessTransaction name="Clipping Request">
      - <RequestingBusinessActivity name="">
          <DocumentEnvelope isPositiveResponse="true" businessDocument="Clipping Request" />
        </RequestingBusinessActivity>
      - <RespondingBusinessActivity name="">
          <DocumentEnvelope isPositiveResponse="true" businessDocument="Clipping List" />
        </RespondingBusinessActivity>
      </BusinessTransaction>
    + <BusinessTransaction name="Create Order">
    - <BusinessTransaction name="Notify Creation">
      - <RequestingBusinessActivity name="">
          <DocumentEnvelope isPositiveResponse="true" businessDocument="ClipList" />
```

ebXML - CPPs und CPAs

- *Collaboration Protocol Profiles*
  - Damit Handelspartner miteinander kommunizieren können, muss bekannt sein, was die Partner können und was nicht.
  - CPP identifiziert den Business Prozess, an dem die Handelspartner beteiligt sind, sowie die Rolle der Partner (z.B. **`buyer, insurer`**)
  - Zudem wird ein Transportprotokoll (z.B. HTTP) festgelegt.
  - Falls nötig, wird das Dokument digital signiert.
  - Dann wird das CPP in der ebXML registriert und mit einem GUID (Global Unique Identifier) versehen. Dieser wird Teil der Metadaten (der GUID ist nicht Teil des Dokuments).
  - Mögliche Handelspartner können die CPPs absuchen und potentielle Partner ausfindig machen.

ebXML - CPPs und CPAs

- *Struktur einer CPP*
  - CPP‘s besitzen als Root **`CollaborationProtocolProfile`** und **`PartyInfo`**, **`Packaging`**, **`Signature`** und **`Comment`** Elementen
    -
```
<CollaborationProtocolProfile
      xmlns="http://www.ebxml.org/namespaces/tradePartner"
      xmlns:ds="http://www.w3.org/2000/09/xmldsig#“
      xmlns:xlink=http://www.w3.org/1999/xlink
      version="1.1">
      <PartyInfo> ...
      <!--REQUIRED, Repeatable--> ...
      </PartyInfo>
      <Packaging id="ID"> ... <!--REQUIRED--> ...
      </Packaging>
      <ds:Signature> ... <!--OPTIONAL--> ...
      </ds:Signature> <Comment> ...
      <!-- OPTIONAL --> ...
      </Comment>
</CollaborationProtocolProfile>
```

ebXML - CPPs und CPAs

- *Struktur einer CPP*
  - Das Root Element CollaborationProtocolProfile benötigt drei Namensraum-Deklarationen:
    - **`http://www.ebxml.org/namespaces/tradePartner`**
      Dies ist der Default ebXML Namensraum..
    - **`http://www.w3.org/2000/09/xmldsig#`**
      ist der Namesraum für XML Digitale Signaturen und wird benötigt, damit die CPPs signiert werden können.
    - **`http://www.w3.org/1999/xlink`**
      Dieser XLink Namesraum gestattet es den CPPs auf externe Referenzinformationen zu verweisen.
  - Für das Dokument selber sind lediglich die Elemente
    `PartyInfo` und **`Packaging`**
    zwingend erforderlich

ebXML - CPPs und CPAs

- *Das* **`PartyInfo`** *Element*
  - Das **`PartyInfo`** Element enthält Informationen über die Organisation.
    - Falls die Organisation aus mehreren Teilen besteht, können mehrere **`PartyInfo`** Elemente verwendet werden.
  - Das **`PartyInfo`** Element umfasst:
    - Ein oder mehrere **`PartyId`** Elemente.
      - Diese Elemente liefern eine logische Adresse für die Organisation, beispielsweise eine DUNS Identifikation.
    - Ein **`PartyRef`** Element.
      - Dieses Element zeigt auf externe Ressourcen mit weiteren Informationen zur Organisation.
    - Ein oder mehrere **`CollaborationRole`** Elemente.
      - Diese Elemente bilden das Kernstück eines CPPs.
      - Sie enthalten Informationen über Business Prozesse, an denen die Organisation beteiligt ist, sowie über die Rolle, welche die Organisation in diesem Prozess spielt.
      - Das **`CollaborationRole`** Element referenziert eine *Business Process Specification* in der Registry.

ebXML - CPPs und CPAs

- *Das* **`PartyInfo`** *Element*
  - Das **`PartyInfo`** Element umfasst:
    - Ein oder mehrere **`Certificate`** Elemente.
      - Diese Elemente identifizieren das Security Certificate der Teilnehmer.
    - Ein oder mehrere **`DeliveryChannel`** Elemente.
      - Diese Elemente definieren die Art und Weise, in der ein Partner Messages empfangen kann (inklusive Dokumentaustausch, Message Protokoll und Transport Protokoll Layer)
    - Ein oder mehrere **`Transport`** Elemente.
      - Diese Elemente liefern Details zu den Transport Layern (HTTP, SMTP oder andere Transport Protokolle).
    - Ein oder mehrere **`DocExchange`** Elemente.
      - Diese Elemente enthalten spezifische Informationen über die Dokumente, welche ausgetauscht werden und im **`DeliveryChannel`** Element referenziert werden.

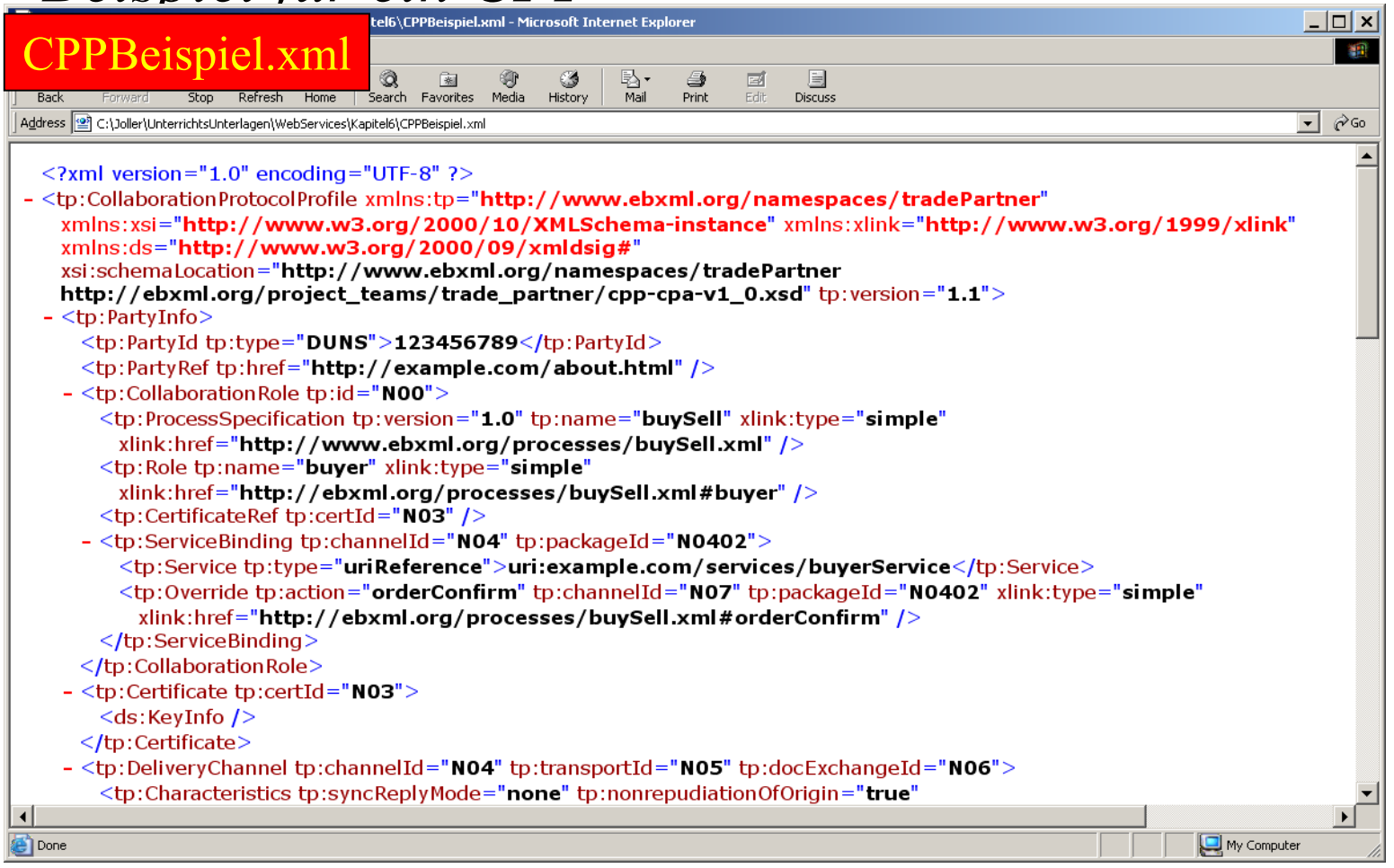
ebXML - CPPs und CPAs

- *Der Aufbau einer Message* - **`Packaging`**
  - Das **`Packaging`** Element enthält Informationen über die Art und Weise, in der Messages aufgebaut sind.
  - Messages werden als SOAP Messages mit Attachments betrachtet und das **`Packaging`** Element liefert Informationen über die Art und Weise, wie diese Messages organisiert sind.
  - Das **`Packaging`** Element besitzt drei potentielle Child Elemente:
    - Das **`ProcessingCapabilities`** Element ist leer, hat aber zwei zwingende Attribute **`generate`** und **`parse`**, welche anzeigen, ob das System Messages kreieren oder lesen kann.
    - Das **`SimplePart`** Element definiert Message Stücke, welche zusammen einem bestimmten Multipurpose Internet Mail Extenstions (MIME) Type entsprechen. Die Teile werden eindeutig gekennzeichnet, so dass sie im **`CompositeList`** Element referenziert werden können.
    - Das **`CompositeList`** Element enthält Informationen über die Komposition oder Kapselung von **`SimpleParts`**.
    - Dieses Element ist optional.

ebXML - CPPs und CPAs

- *Beispiel für ein CPP*

CPPBeispiel.xml



ebXML - CPPs und CPAs

- *Das Collaboration Protocol Agreement (*CPA*)*
  - Die CPA ist in mancher Hinsicht die Schnittstelle zweier CPP‘s.
  - Beispiel:
    - Falls ein Partner etwas verkaufen will
      und der andere Partner etwas kaufen möchte
      dann haben wir perfektes Zusammenspiel.
    - Andernfalls müssen die Partner die Details der Zusammenarbeit aushandeln.
  - Nachdem beide Partner sich geeinigt haben, besitzen beide identische Kopien der gleichen CPA.
  - Diese wird eingesetzt, um ihr jeweiliges System passend zu konfigurieren.
  - CPA‘s können auch in die Registry eingetragen werden, aber dies ist nicht zwingend.

ebXML - CPPs und CPAs

- *Struktur einer CPA*
  - Die Struktur einer CPA ist der einer CPP ähnlich.
    - **<CollaborationProtocolAgreement**

```
xmlns="http://www.ebxml.org/namespaces/
       tradePartner"
xmlns:ds = "http://www.w3.org/2000/09/xmldsig#"
xmlns:xlink =“http://www.w3.org/1999/xlink”
cpaid=“http://www.example.com/cpas/clipCPA”
version="1.7">
<Status value = "proposed"/>
<Start>1988-04-07T18:39:09</Start>
<End>1990-04-07T18:40:00</End>
<ConversationConstraints invocationLimit = "250“
       concurrentConversations = "5"/>
<PartyInfo><!--REQUIRED,repeatable--></PartyInfo>
<Packaging id="N20"> <!--REQUIRED, repeatable-->
</Packaging>
<ds:Signature> <!--OPTIONAL--></ds:…>
</CollaborationProtocolAgreement>
```

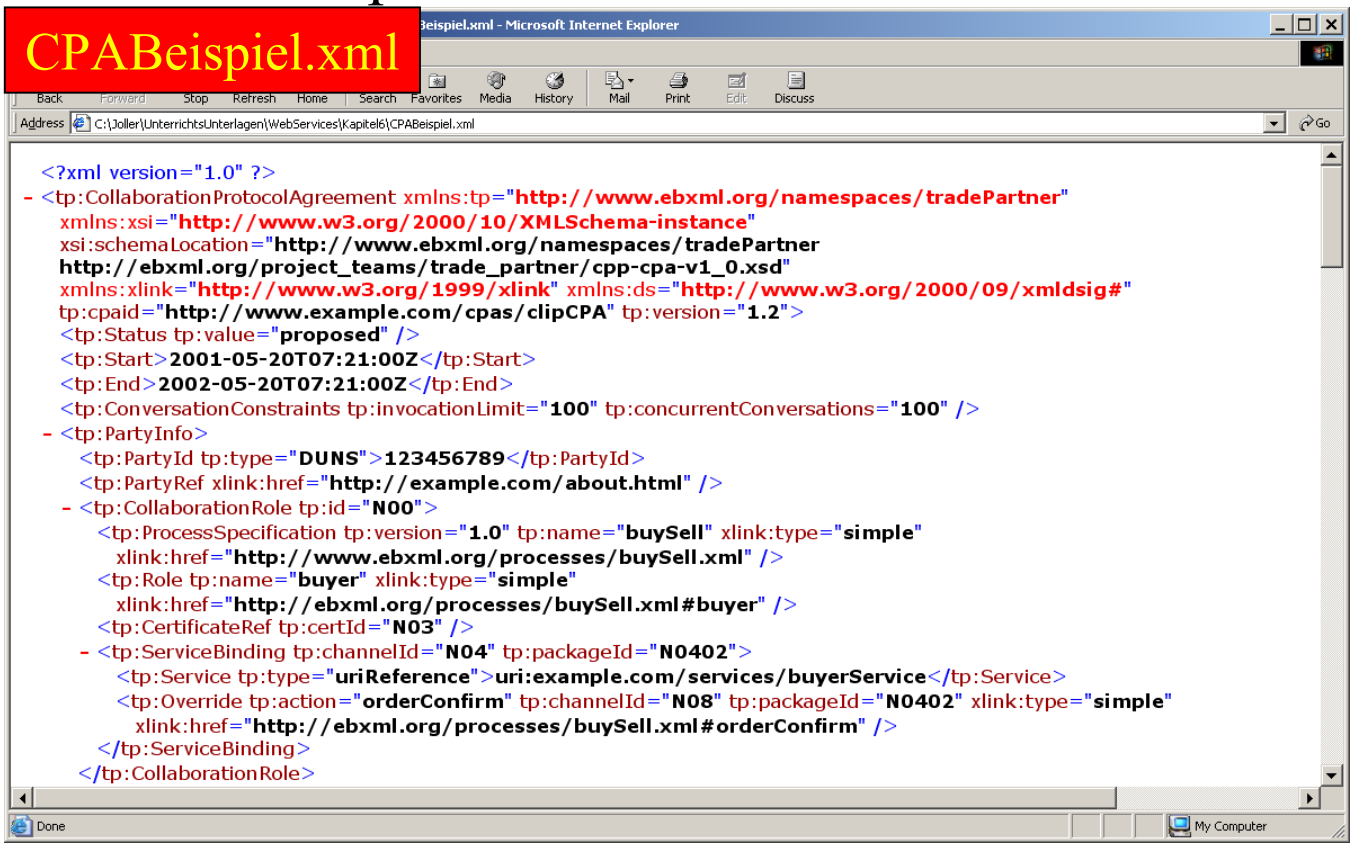
ebXML - CPPs und CPAs

- *Struktur einer CPA*
  - Wie bei der CPP werden zuerst Namensräume definiert, sowie eine Versionsnummer.
  - Das **`cpaid`** Attribute kann eingesetzt werden, um eine URI anzugeben.
    - Die Bedeutung der Elemente **`PartyInfo`**, **`Packaging`**, **`Signature`** und **`Comment`** ist die selbe wie beim CPP, ausser dass **`PartyInfo`** Elemente beider Partner in der CPA enthalten sind.
    - Typischerweise generiert ein Partner eine CPA und offeriert diese dem anderen Partner. Das Status Element zeigt den **`Status`**: **`proposed`**, **`agreed`** oder **`signed`**.
    - Das **`Start`** und **`End`** Elemente repräsentieren, in Coordinated Universal Time, Start und Ende während der die CPA aktiv ist.
    - Das optionale Element **`CoversationConstraints`** definiert die Anzahl *Konversationen,* welche unter diesem CPA stattfinden können, sowie die Anzahl gleichzeitiger Konversationen.

ebXML - CPPs und CPAs

- *Beispiel für eine CPA*
  - Dieses Beispiel basiert auf dem CPP weiter vorne!

CPABeispiel.xml



ebXML - Die Registry

- *Die ebXMLRegistry ist das Kernstück, ist aber noch unvollständig.*
  - OASIS ebXML v2 ist die Registry Reference Implementation.
  - Ihre Web Adresse

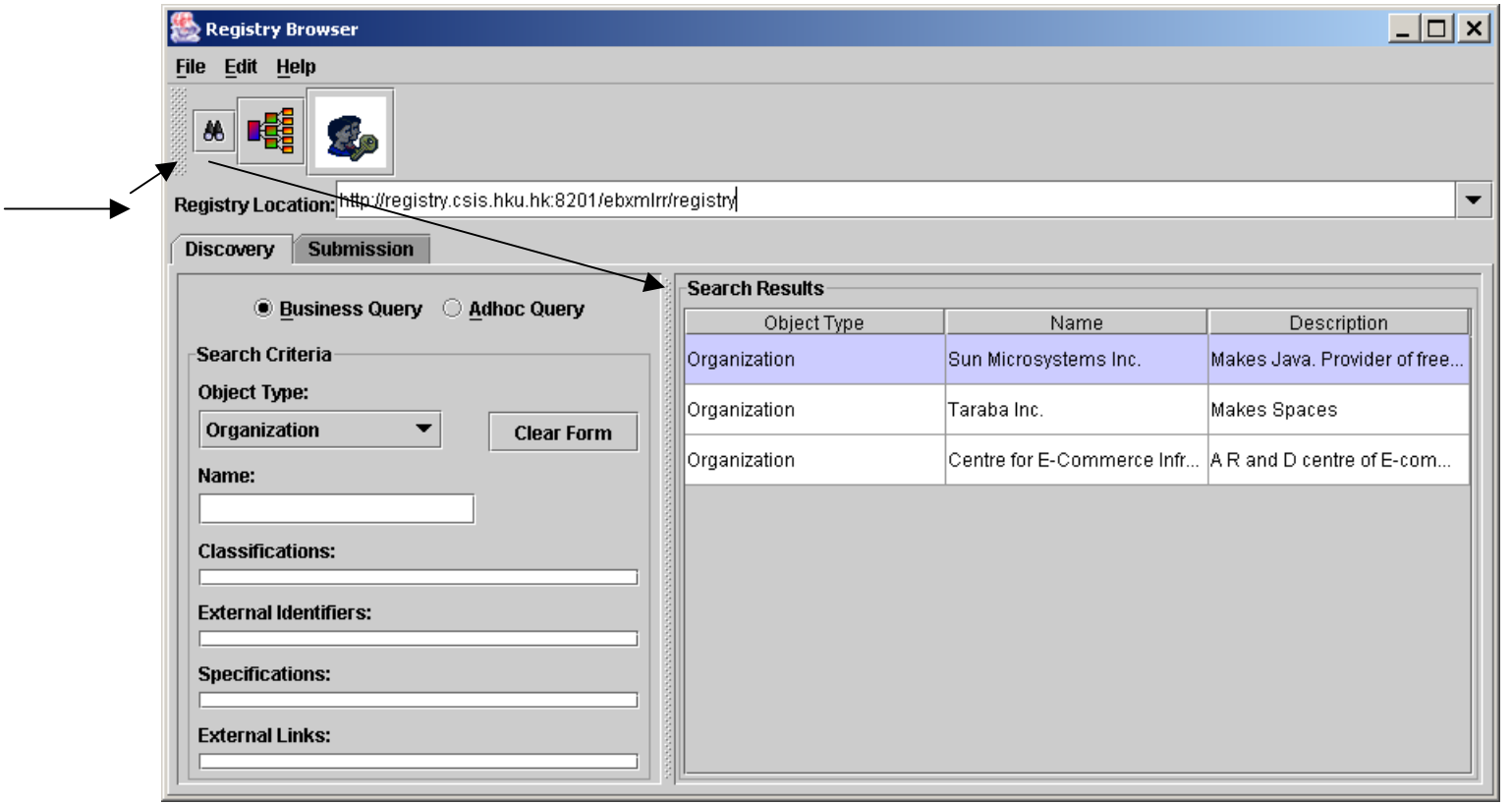
    http://ebxmlrr.sourceforge.net/registryBrowser/registryBrowser.jnlp.
    - Diese Adresse anklicken oder im Browser starten, dann öffnet Java Web Start und lädt die erforderlichen Komponenten herunter und startet den Registry Browser
      (in Java Web Start muss Java 1.4 konfiguriert sein).

ebXML - Die Registry

- *Abfrage der Registry – mit Web Start*
  - Web Start lädt jede Menge Software herunter, u.a. XALAN.jar.
  - Das Herunterladen dauert (ca 15 MB).
  - Wählen Sie eine der Registries aus
    - Ihnen steht ein Pull-Down Menu zur Verfügung.
    - Suchen Sie einfach die gesamte Registry ab, ohne Suchwort.
    - Der Feldstecher oben links startet die Suche.
  - Ergebnis:
    - *Organisationen*
    - Klicken Sie auf eine, so öffnet sich ein weiteres Fenster mit den Details.

ebXML - Die Registry

- *Abfrage der Registry – mit Web Start*

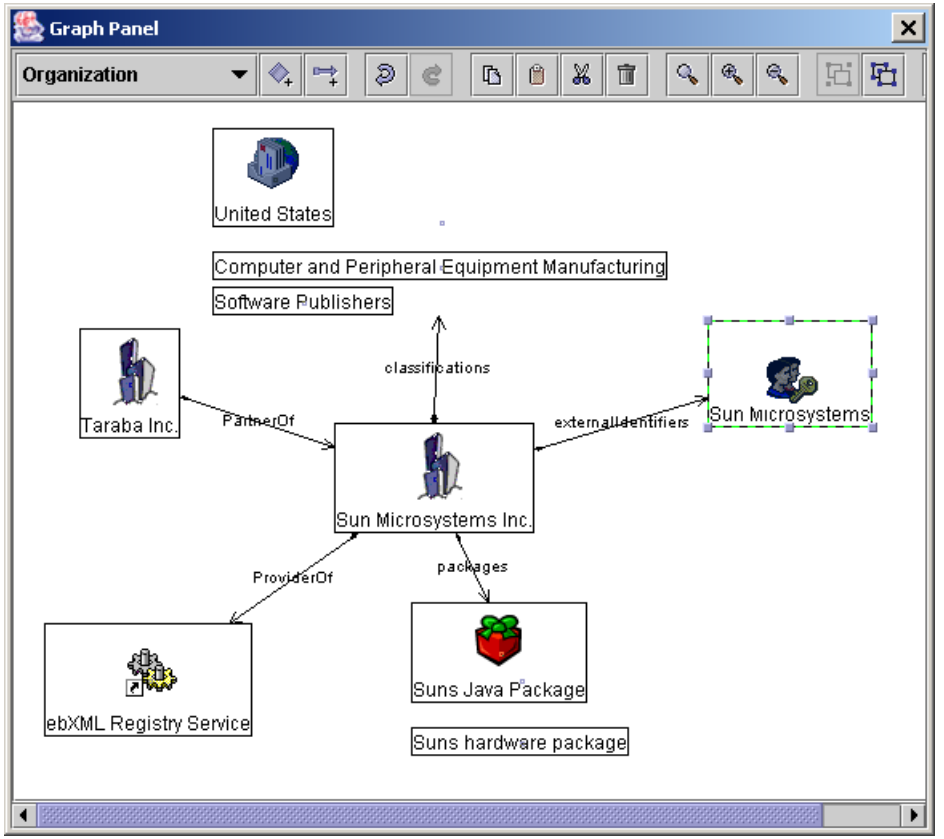


ebXML - Die Registry

- *Assoziation in der Registry abfragen*
  - Wählen Sie Sun
  - Klicken Sie auf die rechte Maustaste:
    - Wählen Sie **`BrowseRegistryObject`**
  - Es öffnet sich ein weiteres Fenster mit einem Sun Icon
    - Wählen Sie das Icon an und
  - Klicken Sie auf die rechte Maustaste:
    - Wählen Sie Show Related Objects
    - Nun sehen Sie einen Teil der Registry Beziehungen.

ebXML - Die Registry

- *Assoziation in der Registry abfragen*



ebXML - Message Service

- *ebXML Message Service (ebMS) ist zentral für ebXML*
  - Jede ebXML Message besteht aus einer Anzahl MIME Bestandteile.
    - Der erste Teil ist eine SOAP Message, welche die Message identifiziert.
    - Der Rest ist die Nutzlast der Message.
  - Ein *Message Service Handler* agiert oberhalb der aktuellen Applikation, dem *Message Service Interface* (MSI).
  - ebMS spezifiziert kein Protokoll für das Ausliefern von Messages.
    - HTTP ist das am meisten verbreitete Protokoll
    - Aber auch SMTP
    - Oder IIOP sind einsetzbar.

ebXML - Message Service

- *Die SOAP Message*
  - Der Message Header besteht aus einer SOAP Message mit allen relevanten Informationen über die Message.
  - Die SOAP Message besteht aus einem Umschlag, dem Header sowie dem Rumpf.
  - CPA Informationen stehen im SOAP Header
  - Beispiel:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope
xmlns:xlink="http://www.w3.org/1999/xlink" ... >
<SOAP:Header xmlns:eb= "http://www.oasis-
open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd"
... > ... </SOAP:Header>
<SOAP:Body xmlns:eb= "http://www.oasis-
open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd"
... > ... </SOAP:Body>
</SOAP:Envelope>
```

ebXML - Message Service

Der SOAP Header

```
<?xml version="1.0" encoding="UTF-8"?> <SOAP:Envelope
">
<eb:MessageHeader SOAP:mustUnderstand="1“
 eb:version="2.0">
<eb:From> <eb:PartyId>urn:duns:987654321</eb:PartyId>
</eb:From>
<eb:To> <eb:PartyId>urn:duns:123456789</eb:PartyId>
</eb:To><eb:CPAId>http://www.example.com/cpas/clipCPA
</eb:CPAId><eb:ConversationId>
http://www.example.com/cpas/clipCPA@25430
</eb:ConversationId>
<eb:Service>urn:services:ClipProcessing</eb:Service>
<eb:Action>OrderClips</eb:Action> <eb:MessageData>
<eb:MessageId>http://www.example.com/cpas/clipCPA@25430
@648124</eb:MessageId>
<eb:Timestamp>2001-05-16T12:01:00</eb:Timestamp>
</eb:MessageData> <eb:DuplicateElimination/>
</eb:MessageHeader> </SOAP:Header> <SOAP:Body > ...
</SOAP:Body> </SOAP:Envelope>
```

ebXML - Message Service

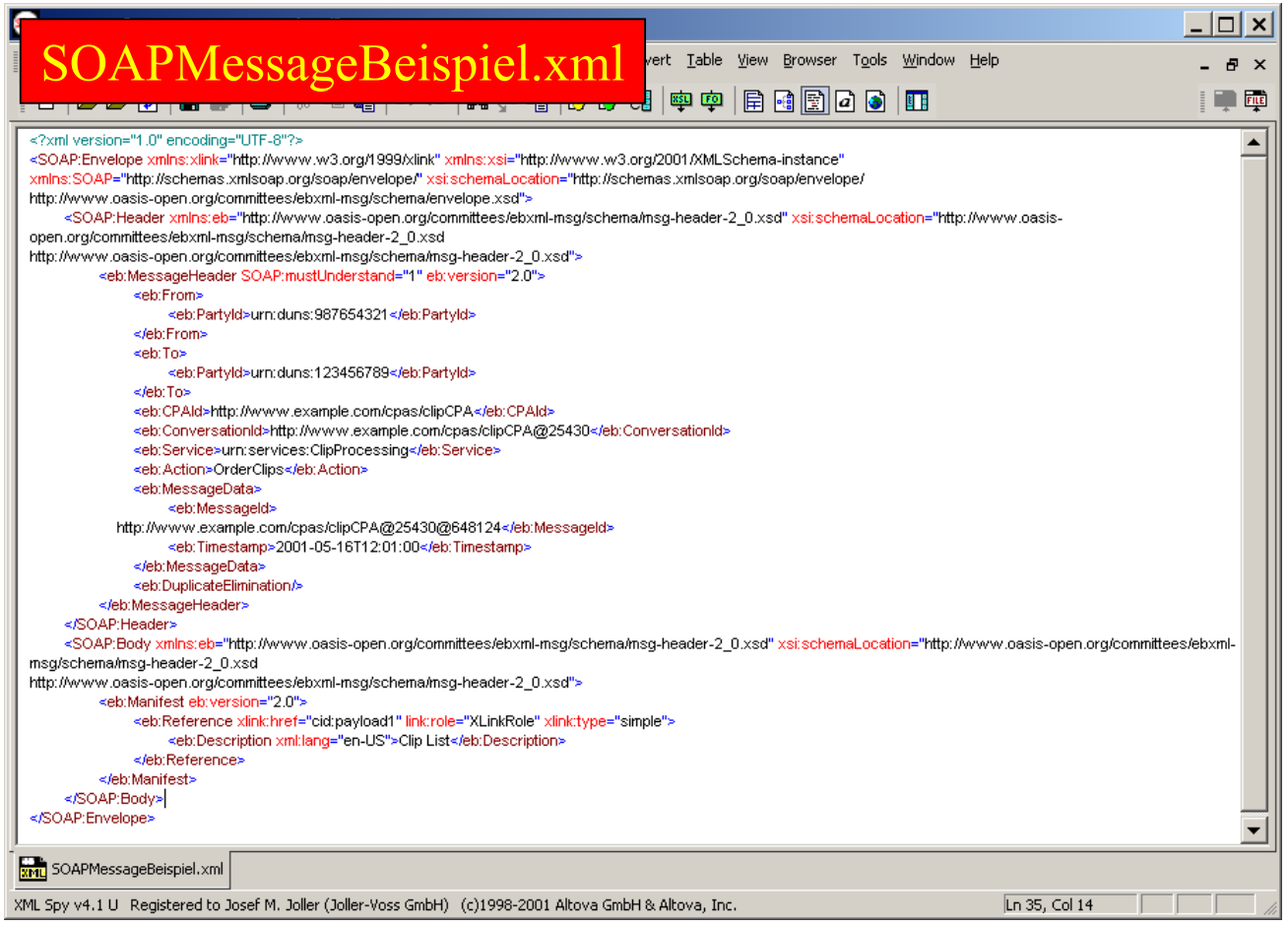
- *Der SOAP Body*
  - <?xml version="1.0" encoding="UTF-8"?>
    <SOAP:Envelope… > <SOAP:Header …> ...
    </SOAP:Header>
    <SOAP:Body …>
    <eb:Manifest eb:version="2.0">
    <eb:Reference xlink:href="cid:payload1"
    xlink:role="XLinkRole" xlink:type="simple">
    <eb:Description xml:lang="en-US">Clip List
    </eb:Description>
    </eb:Reference>
    </eb:Manifest>

ebXML - Message Service

- *Das Ganze : Eine MIME Message*
  - **Content-type: multipart/related; boundary="boundaryValue"; type="text/xml"; start="<ebxmlheader@example.com>" -- boundaryValue Content-ID: <ebxmlheader@example.com> Content-Type: text/xml**
    <?xml version="1.0" encoding="UTF-8"?>
    <SOAP:Envelope …>…
    </SOAP:Envelope>
    --boundaryValue
    Content-ID: <payload1>
    Content-Type: text/xml
    <?xml version="1.0" encoding="UTF-8"?>
    <purchase_order> <po_number>1</po_number>
    <part_number>123</part_number>
    <price currency="USD">500.00</price>
    </purchase_order>
    --boundaryValue--

ebXML - Message Service

- *Das Ganze : Eine MIME Message*



ebXML - Zusammenfassung

- *ebXML?*
  - ebXML ist eine Gruppe zusammengehörender Spezifikationen:
    - Analyse der Business Prozesse und Business Dokumente
      - BPSS
    - Dokumentation des Firmenprofils, so dass automatisch Geschäftsfälle eingegangen werden können.
      - CPP
    - Dokument-Transfer zwischen Handelspartners, um gemeinsame Ziele zu erreichen.
      - CPA
    - Messages werden vom Business Service Interface an Partner mithilfe von ebMS versandt.
      - MSH sendet Messages mithilfe von SOAP mit Attachments
      - Eine Message enthält eine SOAP Message und die Dokumente als Payload Attachments.

ebXML - Zusammenfassung

- *Ressourcen*
  - ebXML Spezifikationen
    - http://www.ebxml.org/specs/index.htm.
    - Details über BPSS, Core Components, Registry Information Model, ebMS und andere Spezifikationen von OASIS und UN/CEFACT..
  - UMM
    - http://www.gefeg.com/tmwg/n090r10.htm.
  - UML
    - http://www.omg.org/uml/.
  - W3C's Resource Description Framework
    - http://www.w3.org/RDF/.
  - W3C's Web services
    - http://www.w3.org/2002/ws/.
  - W3C XML Signature Working Group
    - http://www.w3.org/Signature/.