

OMG ARAP - The MDA Approach to a Finance Web Service

Arne J. Berre, SINTEF (Arne.J.Berre@sintef.no)

Todd Boyle, Morten Jacobsen, Netaccount

(tboyle@rosehill.net, mortenjacobsen2000@yahoo.no)

- **OMG GL and ARAP – current standard and submissions**
- **Need for multiple platforms and support for XML (Finance community)**
- **Using MDA: PIM and PSM (EDOC and UML2.0), PIM and GSM**
- **Relating to GSMS: XBRL, EWG and ebXML Core Components**
- **Revised UML model: Using ebXML Core Components basic types**
- **PIM – using the RM/ODP viewpoints**
- **Mappings to CORBA, Web services (WSDL), ebXML, J2EE/EJB**
- **Issues and experiences in PIM and PSM mappings**
- **What should be done next: PIM-PSM mapping standards/tools, 'HUTN XMI'**

For the latest version of this presentation, see: www.arapxml.net

OMG ARAP - The MDA Approach to a Finance Web Service

The OMG **GL** (General Ledger) specification has recently been proposed extended by an **ARAP** (Account Receivable- Account Payable) facility. The initial submission presented in November 2001, was done in the traditional OMG style with CORBA IDL, but the revised submission is currently being done following the **MDA approach through the use of modeling concepts from UML for EDOC and input from the recent UML 2.0 proposals**. The presentation will report on ***experiences and issues*** in doing the platform independent model, and targeting various platforms, including **CORBA, J2EE/EJB, Web services with WSDL and ebXML**.

Accounting Receivable/ Accounting Payable

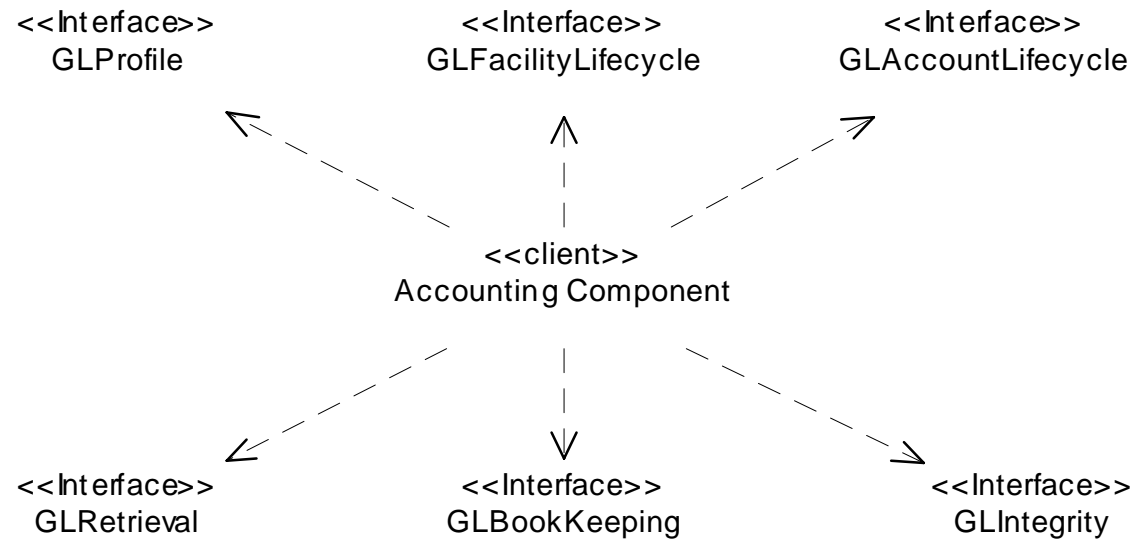
The Account Receivable/Account Payable (AR/AP) Facility defines the interfaces, and their semantics, that are required to enable interoperability between AR/AP systems and general ledgers, sales and purchasing systems, and other distributed objects and applications for accounting purposes.

- **Extension to OMG General Ledger:**

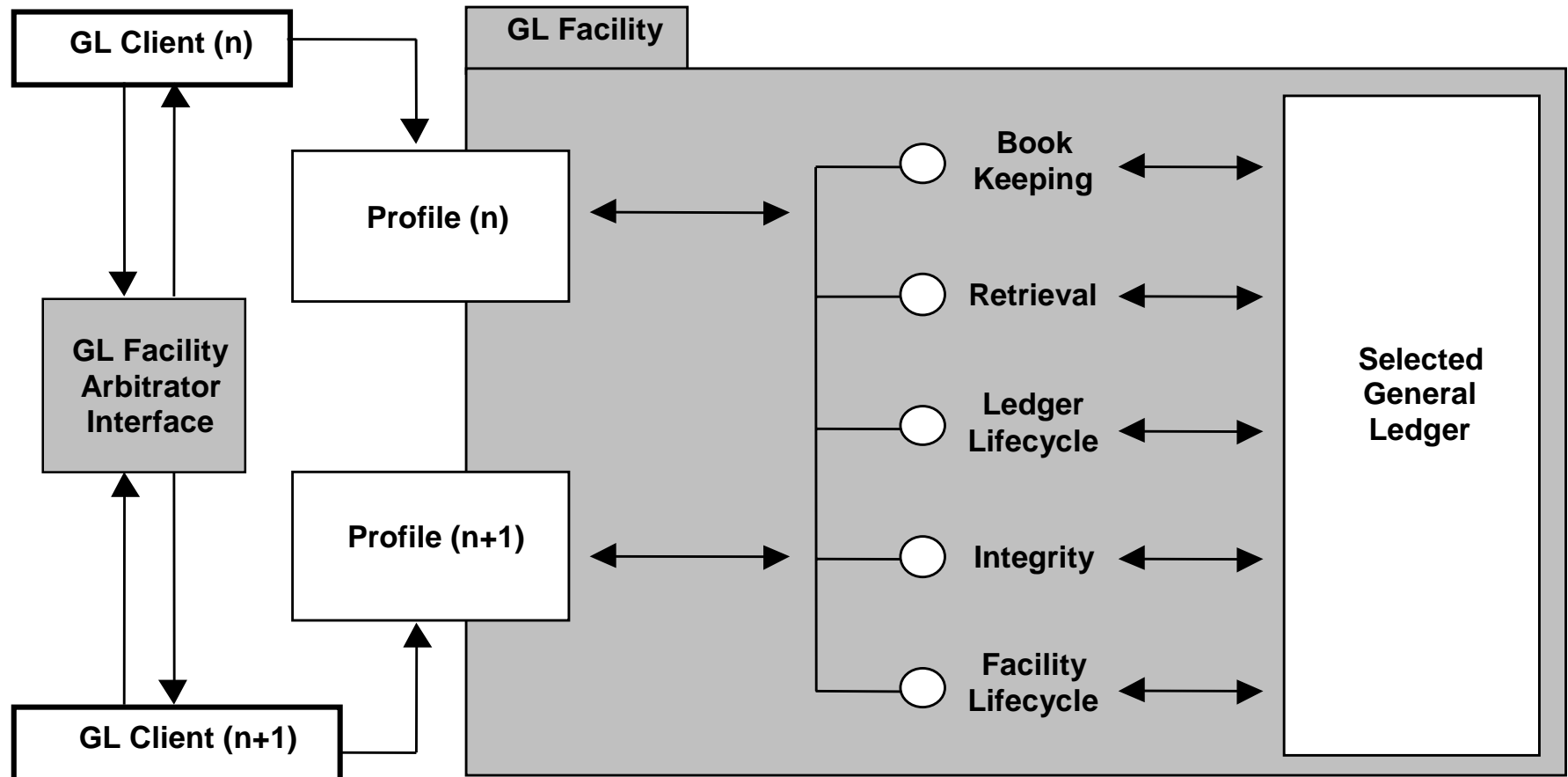
- “Defines the interfaces, and their semantics, that are required to enable interoperability between General Ledger systems and accounting applications, as well as other distributed objects and applications for accounting purposes.”

General Ledger Interfaces

Interface	Purpose	Primary Client(s)
GLProfile	Client Session Establishment	All GL clients
GLBookKeeping	Data entry	Data entry clients
GLRetrieval	Data extraction	Reporting clients
GLAccountLifecycle	Account lifecycle management	GL administration clients
GLIntegrity	Data integrity checks	GL administration clients
GLFacilityLifecycle	GL lifecycle management	GL administration clients

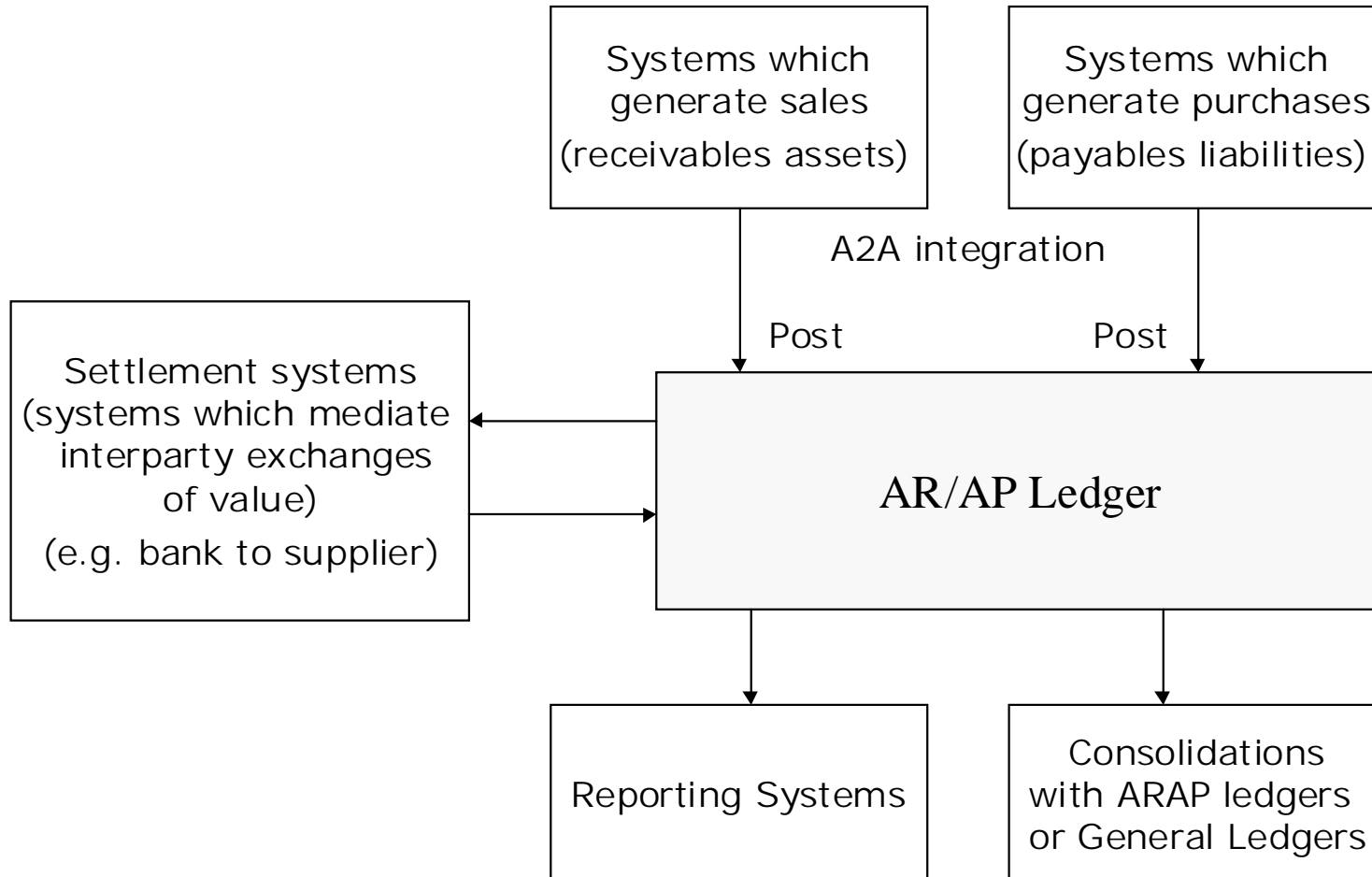


GL Facility - Reference Architecture

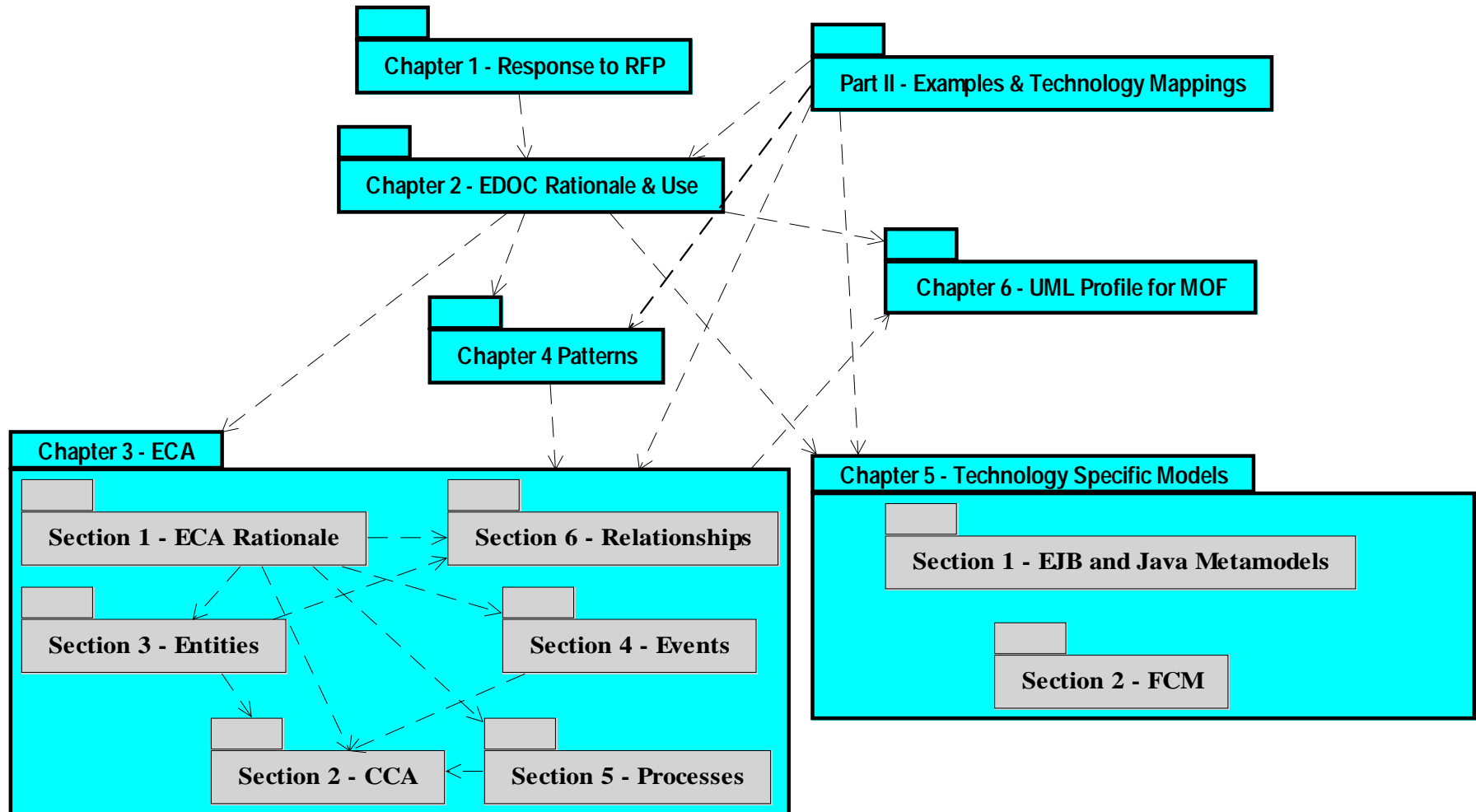


• ... one or more GL clients makes a request for a new GL client session from the GL Facility's Arbitrator interface. If successful, the GL Arbitrator returns a GL Profile interface which provides information about the current session, as well as making provision for controlled access to the various interfaces and operations supported by the GL Facility.

ARAP RFP

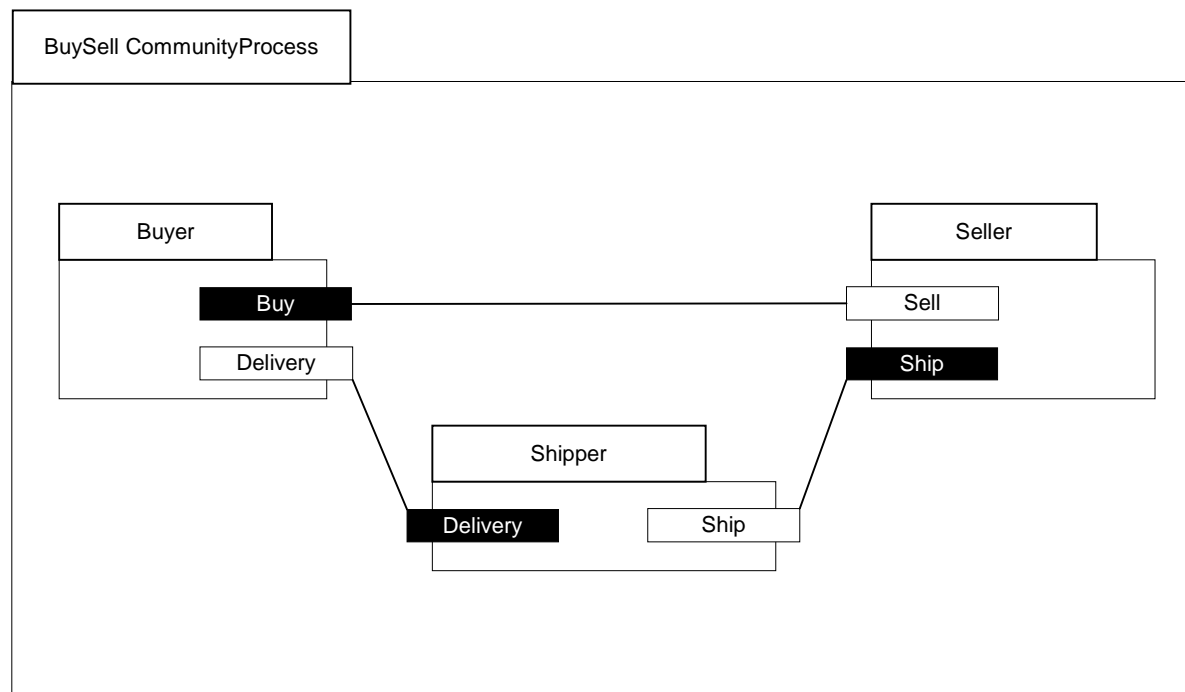


Structure – UML for EDOC



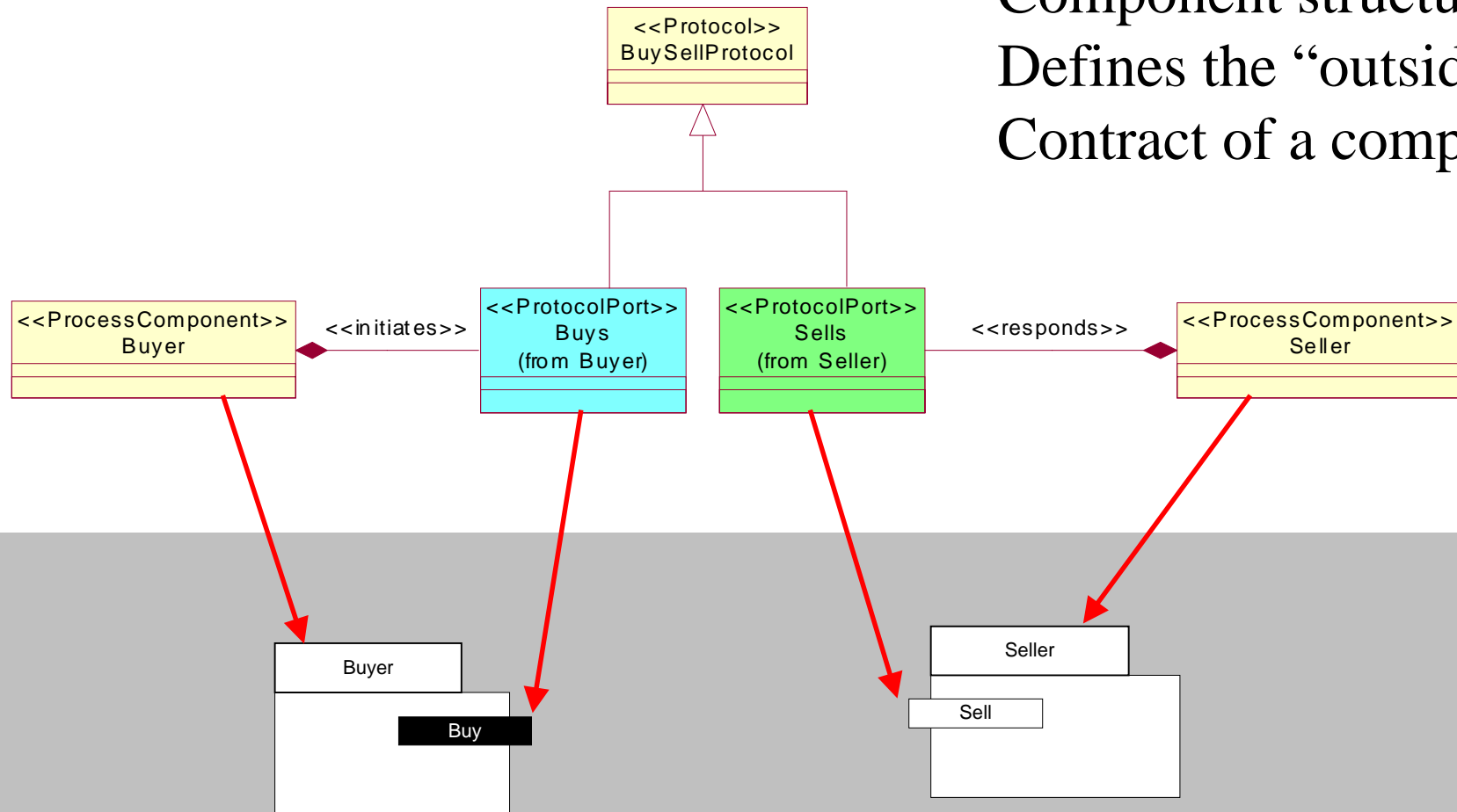
UML for EDOC and UML 2.0 - protocols

- Identify a “community process”, the roles and interactions
- Using CCA Notation

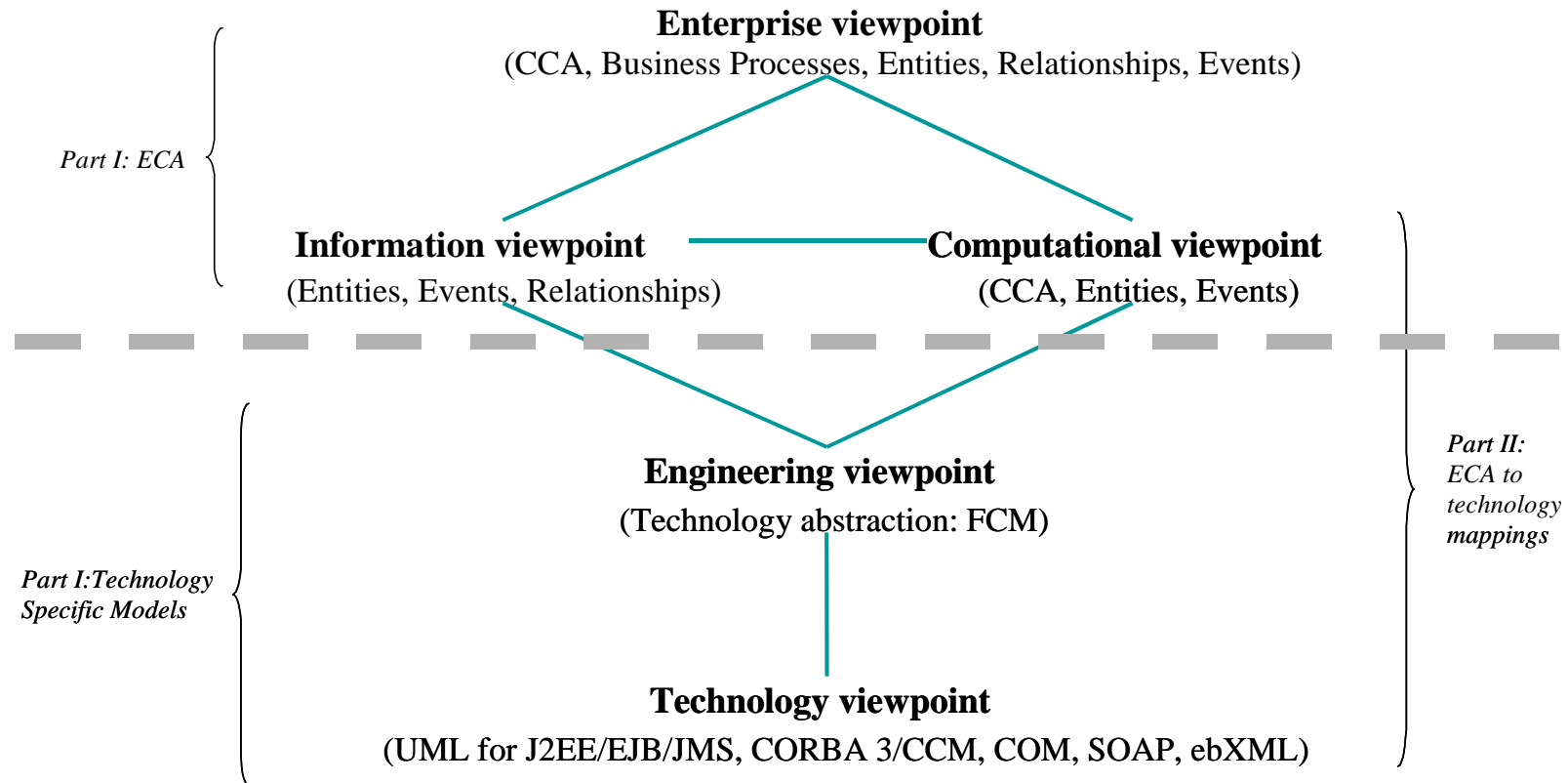


Component structure

Component structure
Defines the “outside”
Contract of a component

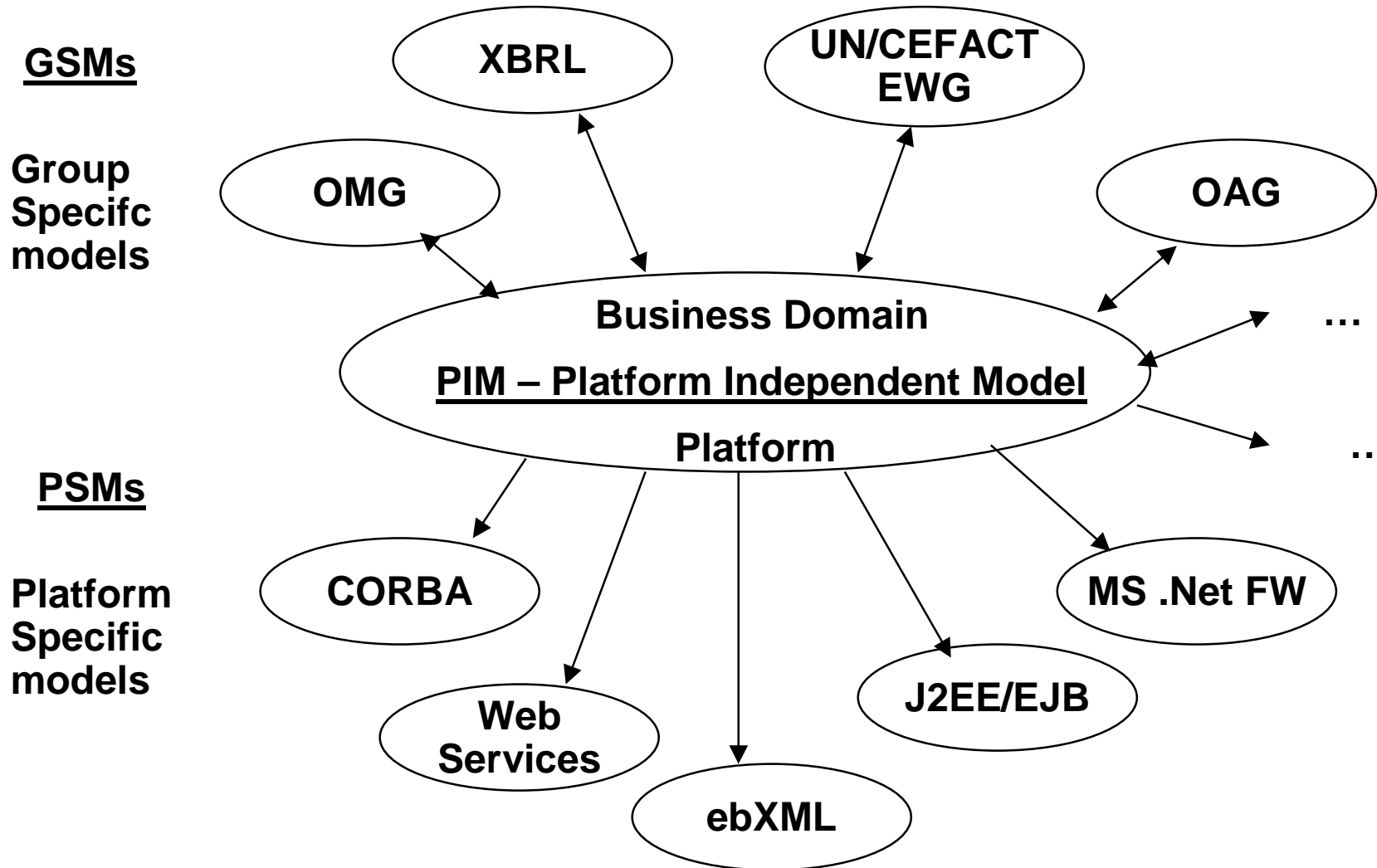


Using RM-ODP viewpoints



Part I: Patterns- applied to all viewpoints

PIM related to PSMs and GSMs



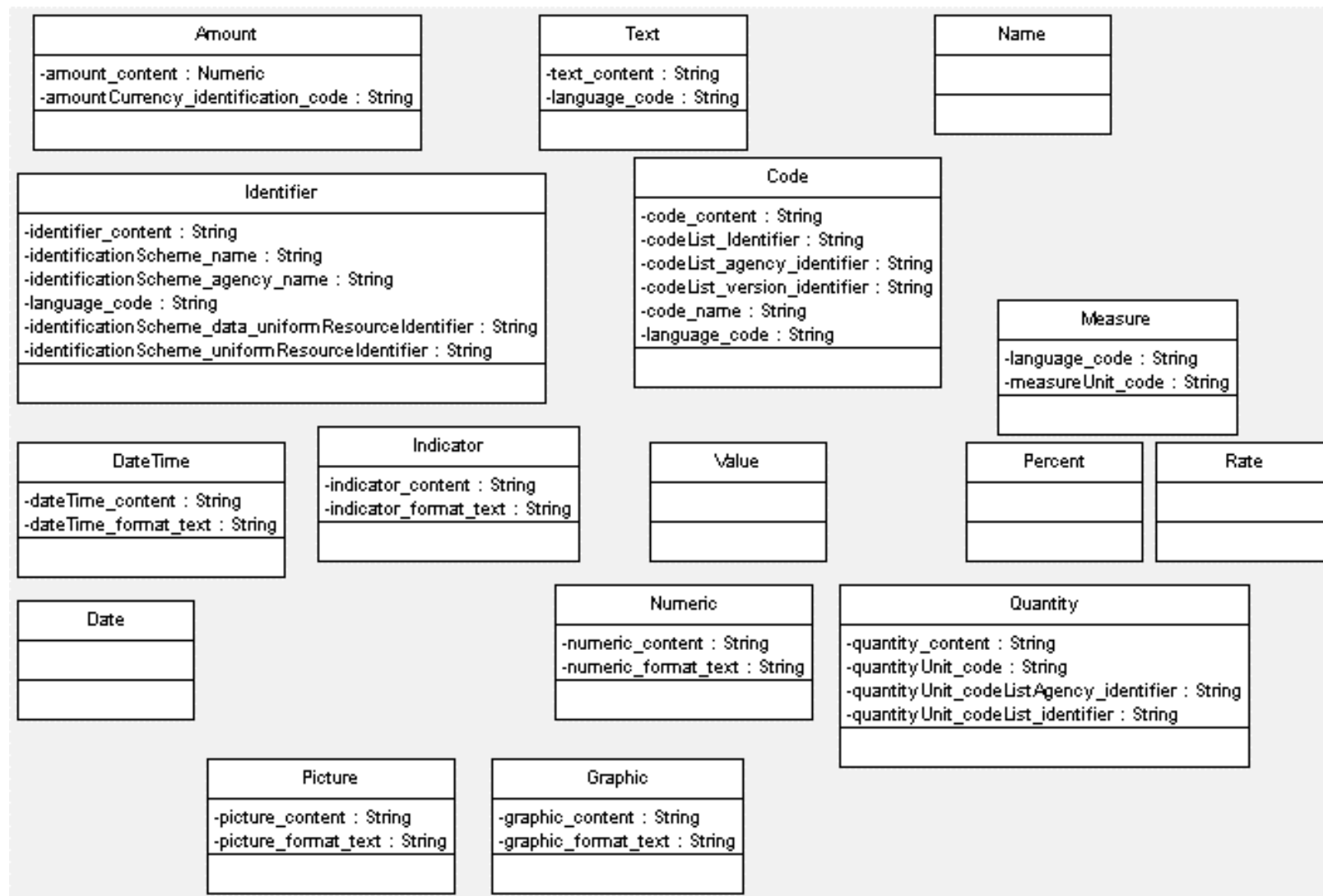
Which basic type system for PIM ?

- **UML for EDOC uses proposal for MOF 1.4**
- **Also need for a library of business types – i.e. CBO – Common Business Objects - OMG has some of those defined with IDL**
- **Other groups has proposed domain based (generic) type systems, i.e. ebXML Core Components, HL7 – Health, ISO 19103 (GIS), ... and their description ISO 11179**

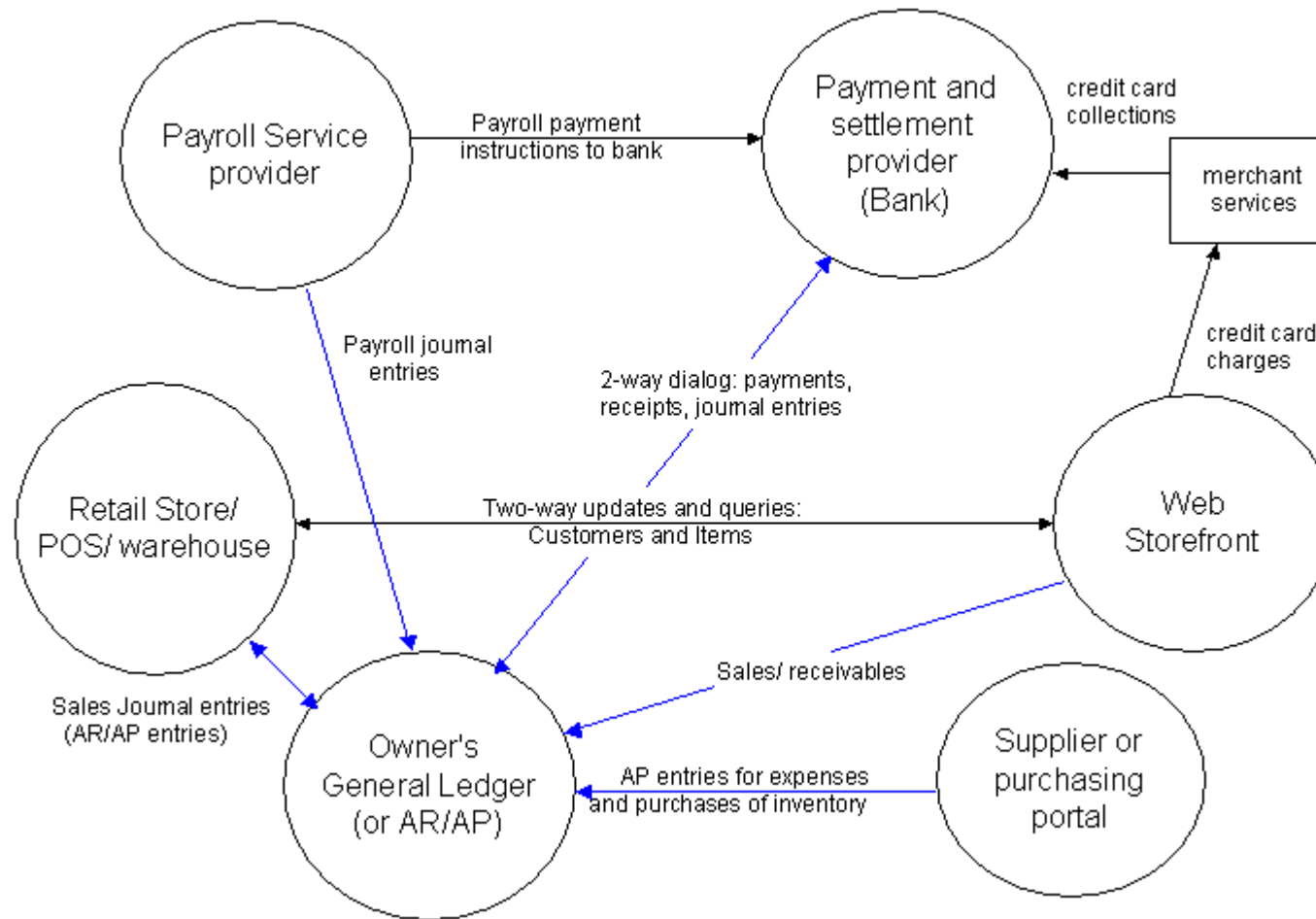
XBRL

- **Extensible Business Reporting Language (XBRL), formerly code-named XFRML, is an open specification which uses XML-based data tags to describe financial statements for both public and private companies. XBRL benefits all members of the financial information supply chain.**
- **Public draft XBRL for General Ledger (XBRL GL)
Summary: The XBRL GL, a way to represent the information that comes into and out from the General Ledger in the international sense**
- **www.xbrl.org**

ebXML Core Component Types



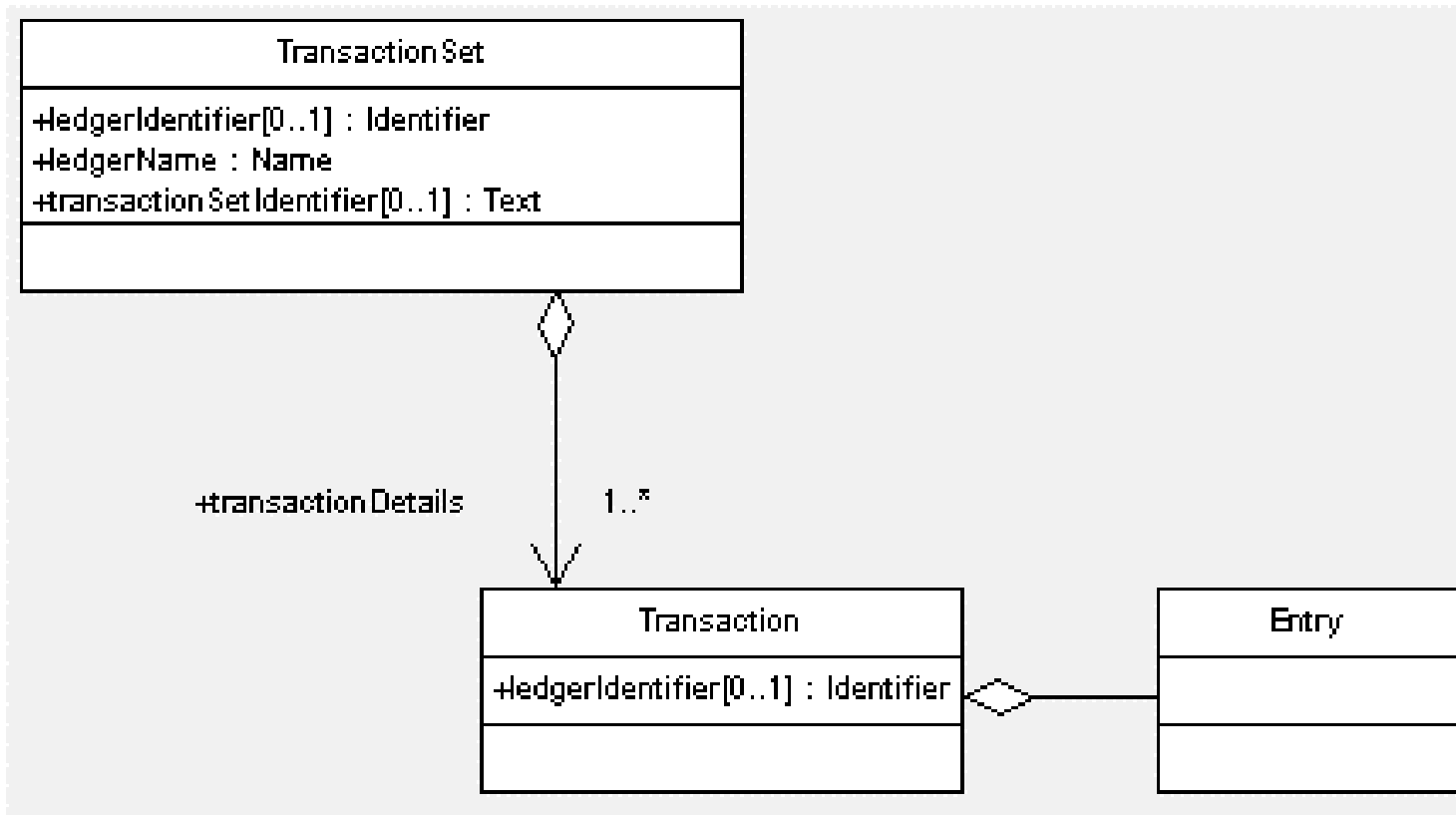
Enterprise viewpoint

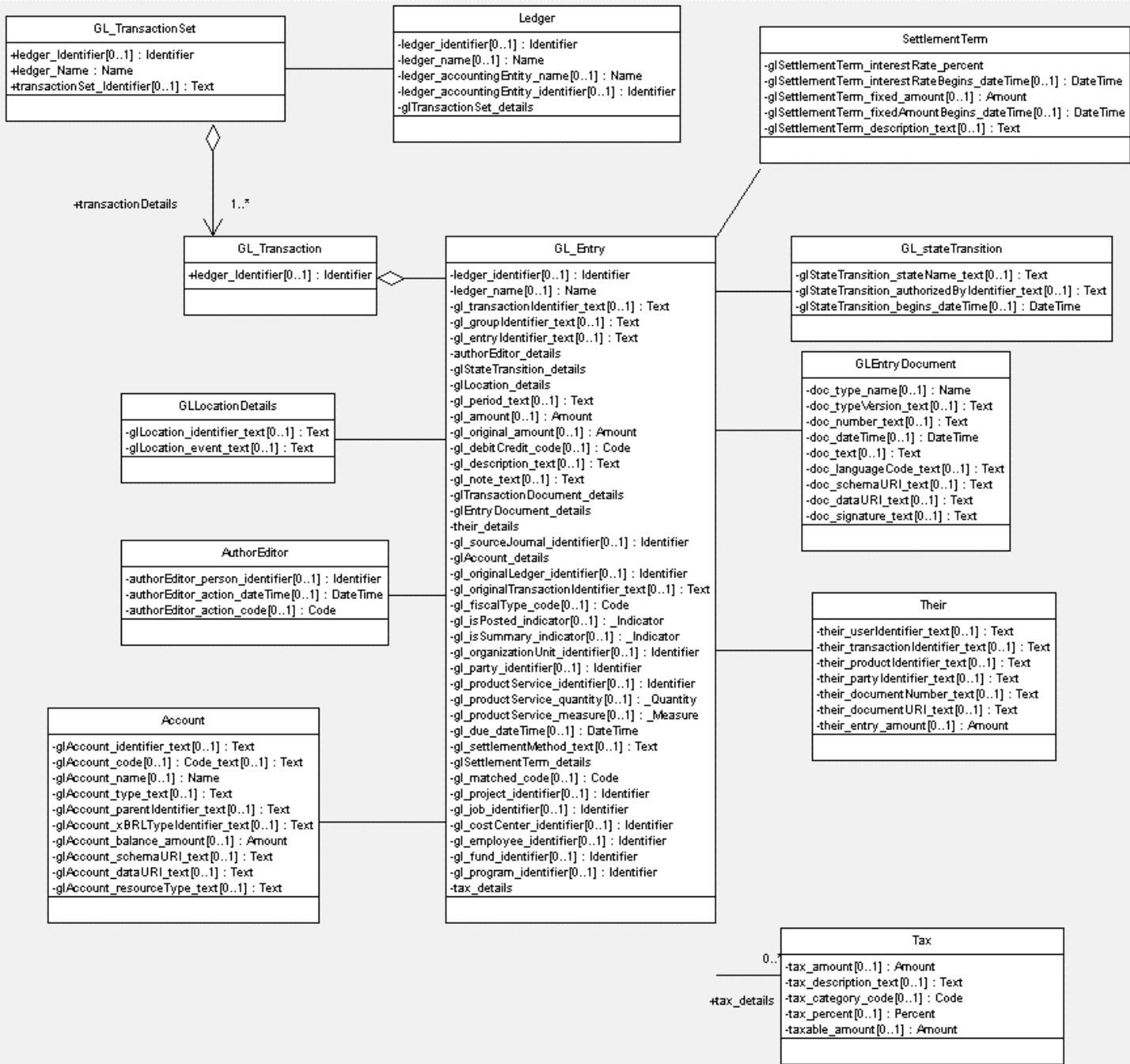


Accounting Patterns & GL/ARAP

OK	1: Single-Entry Bookkeeping	OK	5: Accounting Periods	9: Accounts Receivable/Payable	OK
OK	2: Double-Entry Bookkeeping	Japan =/ west	6: Profit & Loss	10: Dimensions	OK
by rep. codes	3: Account Hierarchy	out	7: Standard Chart of Accounts	11: GL Roll-up	out
Japan =/ west	4: Balance Sheet	Doc.ref	8: Voucher Reference	12: Profit & Loss Roll-Up	Need hierarchy (impl)

Information viewpoint

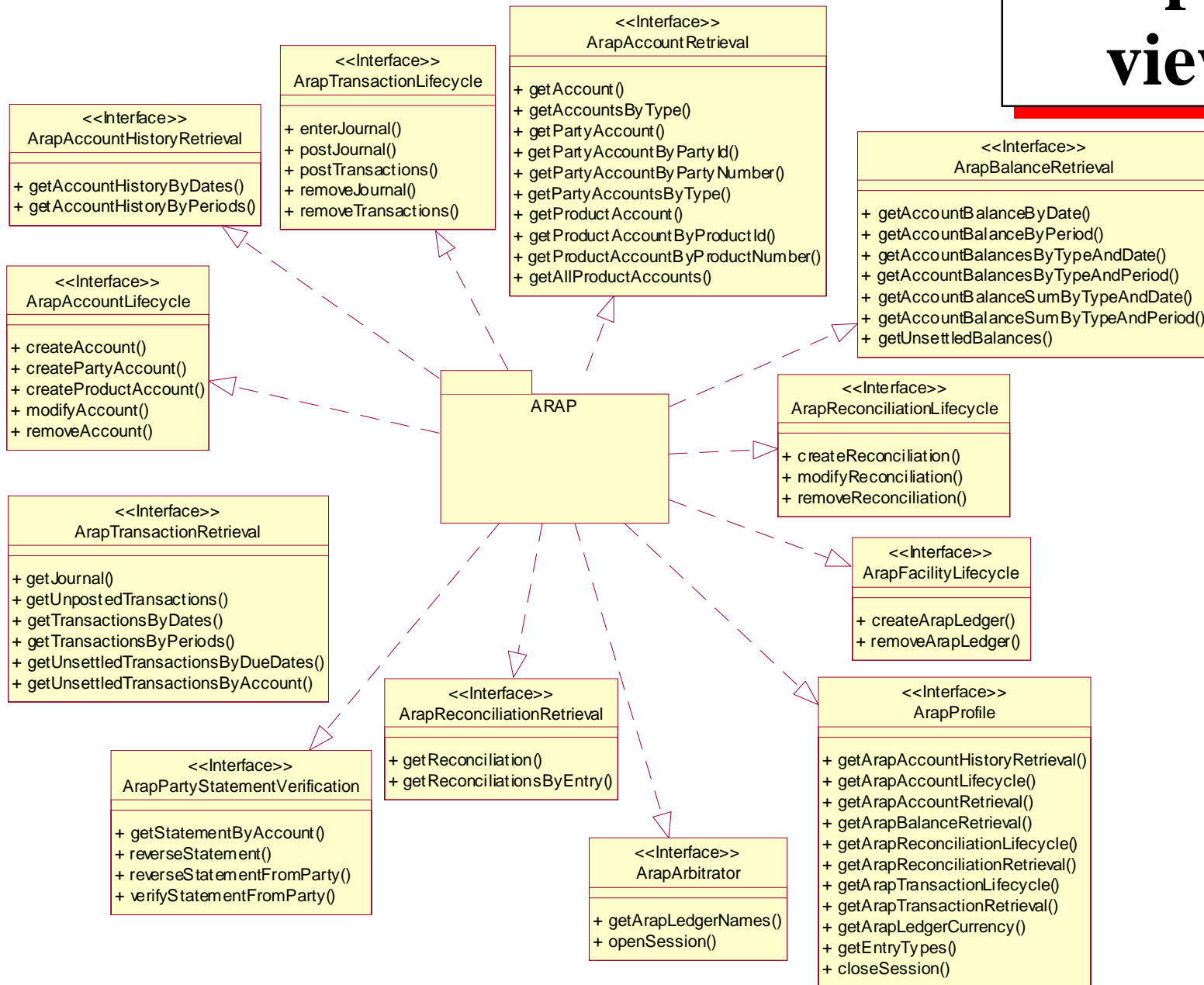




ARAP Transactions and Entry

using ebXML Core Component Types

Computational viewpoint



Engineering viewpoint

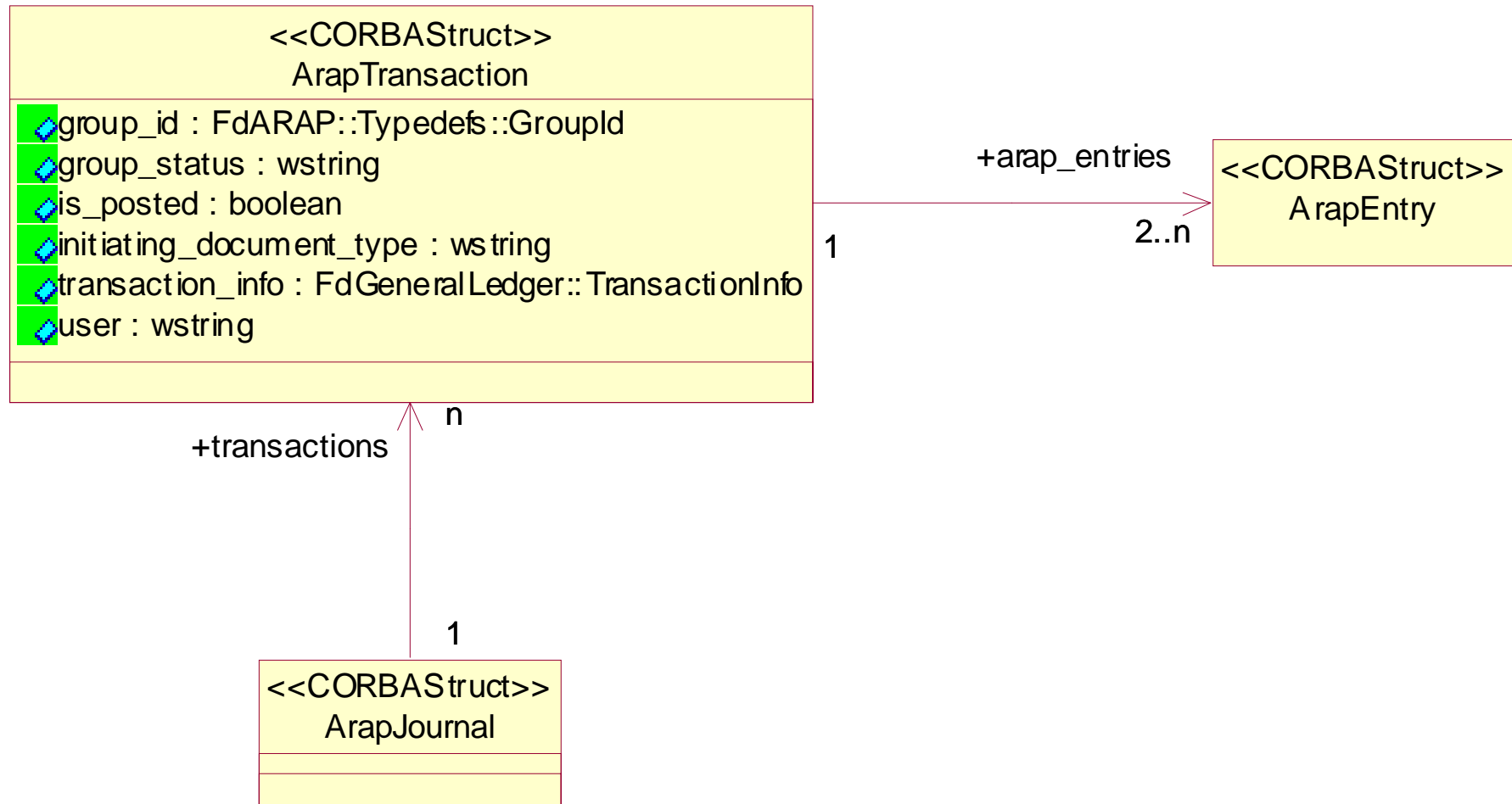
- **How much of this to specify in a PIM ?**
- **Security**
- **Access: Authorisation and Authentication (i.e. Integrating with PKI)**
- **Logging**
- **Distribution transparencies**
- **Internationalization**

Technology viewpoints

- **CORBA**
- **Web Services (WSDL)**
- **ebXML**
- **J2EE/EJB**

- **... (MS .Net Framework)**
- **...**

CORBA PSM - Journal, Transaction, Entry



CORBA PSM - ArapTransactionLifeCycle

<<Interface>>

ArapTransactionLifecycle

```
✦ enterJournal(journal : in FdARAP::Structs::ArapJournal) : FdARAP::Typedefs::JournalId
✦ postJournal(journal_id : in FdARAP::Typedefs::JournalId) : void
✦ postTransactions(transaction_ids : in FdGeneralLedger::TransactionIdList) : void
✦ removeJournal(journal_id : in FdARAP::Typedefs::JournalId) : void
✦ removeTransactions(transaction_ids : in FdGeneralLedger::TransactionIdList) : void
```

XML – Complexity of using XMI

- **We need to be able to generate an XML representation that is looking similar to the “handcrafted” XML that is being made in the finance community**
- **I.e – the “UML HUTN” in XML form – for data exchange, not for model exchange**
- **Will XMI 2.0 be able to do that ? – This is under investigation ...**

Is this improved for XMI 2.0 ?

XML Schema – Basic core component

```
<xsd:complexType name="GLEntry.Details">
```

```
  <xsd:annotation><xsd:documentation>
```

Information about a monetary amount or valuation of an economic event or accountant's adjustment, affecting an accounting entity.

```
</xsd:documentation></xsd:annotation>
```

```
  <xsd:sequence>
```

```
    <xsd:element name="GLLedger.Id" type="GLLedger.Id" minOccurs="0"/>
```

```
    <xsd:element name="GLLedger.Name" type="GLLedger.Name" minOccurs="0"/>
```

```
    <xsd:element name="GL.TransactionId.Text" type="GL.TransactionId.Text" minOccurs="0"/>
```

```
    <xsd:element name="GL.GroupId.Text" type="GL.GroupId.Text" minOccurs="0"/>
```

....

```
<xsd:complexType name="Tax.Amount">
```

```
  <xsd:annotation><xsd:documentation>
```

The amount of tax.

```
</xsd:documentation></xsd:annotation>
```

```
  <xsd:sequence>
```

```
    <xsd:element name="Amount.Content" type="xs:decimal"/>
```

```
    <xsd:element name="AmountCurrency.Identification.Code" type="xsd:string" minOccurs="0" maxOccurs="1"/>
```

```
  </xsd:sequence>
```

```
</xsd:complexType>
```

WSDL

```
<!-- exceptions -->
  <xsd:complexType name="UnknownLedger">
    <xsd:element name="error" type="xsd:string" />
    <xsd:element name="bad_value" type="xsd:string" />

<!-- message types for BookKeeping interface -->
<message name="post_transactionInput">
  <part name="single_transaction" type="tns:Transaction"/>
</message>
<message name="post_transactionOutput">
  <part name="_return" type="tns:TransactionId"/>
</message>
<message name="post_transaction_listInput">
  <part name="transactions" type="tns:TransactionList"/>
</message>
<message name="post_transaction_listOutput">
  <part name="_return" type="tns:TransactionIdList"/>
</message>
```

WSDL

```
<portType name="BookKeepingPort">  
  <operation name="post_transaction" >  
    <input message="post_transactionInput"/>  
    <output message="post_transactionOutput"/>  
    <fault name="fault0" message="PermissionDeniedFault"/>  
    <fault name="fault1" message="BadTransactionFault"/>  
  </operation>  
  <operation name="post_transaction_list">  
    <input message="post_transaction_listInput"/>  
    <output message="post_transaction_listOutput"/>  
    <fault name="fault0" message="PermissionDeniedFault"/>  
    <fault name="fault1" message="BadTransactionInListFault"/>  
  </operation>  
</portType>
```

ebXML PSM

- **No inherent workflow for BPSS – Business Process Specification, General Ledger/ARAP is log of Business Events**
- **Mapping to Business Transactions and Business Document and Messaging**

J2EE/EJB PSM

- Use of JSR-26, UML profile for EJB
- <<EJBImplementation>>
- <<EJBSessionHomeInterface>>
- <<EJBRemoteInterface>>

Experiences and Issues

- **Need to interact with relevant domain groups, and relate to group specific models**
- **Need to have a common basic type system + library of Core components (types)**
- **Need patterns for extension, customisation and flexibility – support for context ?**
- **Need to define principal communication principles (request vs notification oriented) – when to describe synch/asynch (?)**
 - **Pass by value vs reference - vs copy/replication**
 - **Exception modeling vs alternative return messages**
 - **Synchronous vs Asynchronous modeling – message/operation/event**
 - **Identifier (Reference) and CodeList specification and handling**
 - **Need “standard” model transformation rules from PIM to PSM**

Conclusion and further work

- **Platform Independent Models are feasible and useful**
- **Need to define a common set of basic types**
- **Would like to have tool support for the model transformation to platform specific models**

- **Next steps:**
- **Further harmonisation with domain groups (UN/CEFACT EWG meeting, Barcelona in March)**
- **Refining the PIM UML model and applying more detailed model transformations to PIM models for CORBA, Web Services, ebXML and J2EE/EJB**
- **Further information at: www.arapxml.net**