Document Engineering: Designing Documents for Transactions and Web Services

OASIS Symposium - 10 May 2006

Robert J. Glushko (glushko@sims.berkeley.edu)

Who Is This Guy?

- Adjunct Professor at UC Berkeley School of Information since 2002 (www.sims.berkeley.edu/~glushko/)
- Came to I-School from Silicon Valley; founded or co-founded 3 companies in 1990s
- PhD (Cognitive psychology, UCSD) and MS (Software Engineering, Wang Institute) and 25 years in the real world

---

Outline for the Tutorial

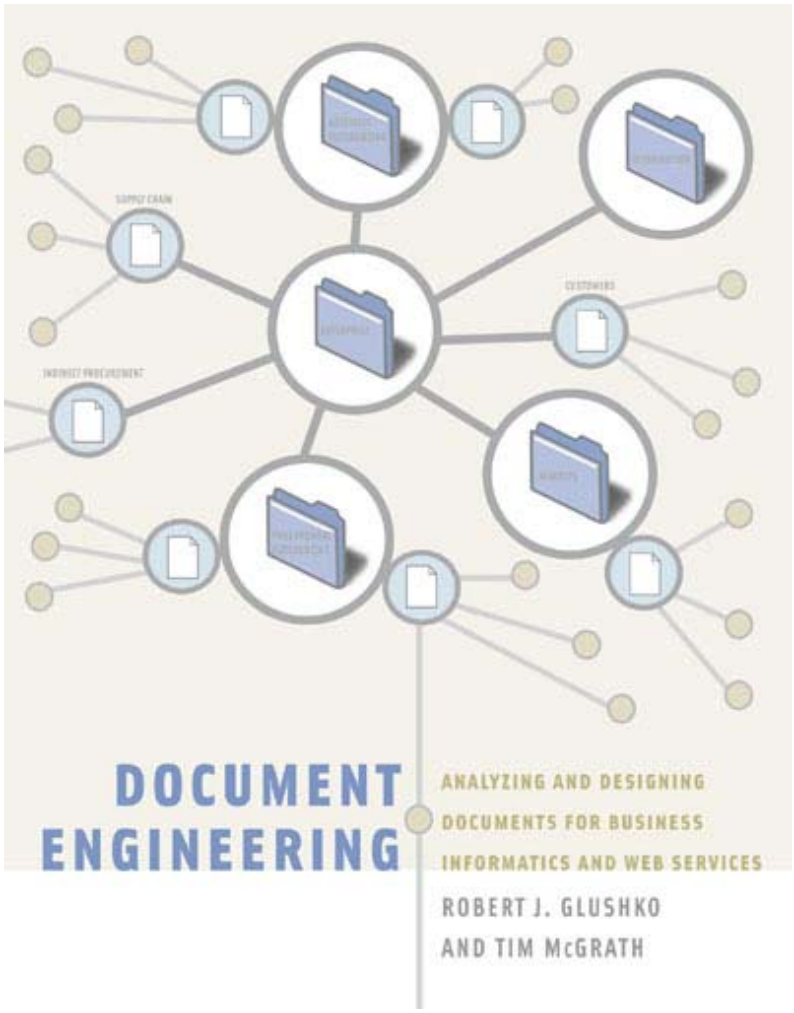
- Document Engineering's Big Ideas
- Document Engineering and XML
- Methods for Modeling Components
- Methods for Modeling Documents

DOCUMENT ENGINEERING'S BIG IDEAS

What is Document Engineering?

- A new discipline for specifying, designing, and implementing the electronic documents that request or provide interfaces to business processes, often via Web-based services
- A synthesis of information and systems analysis, business process modeling, content management, and distributed computing
- A new book (co-authored with Tim McGrath, MIT Press) (http://docengineering.com/)

Document Engineering - The Book

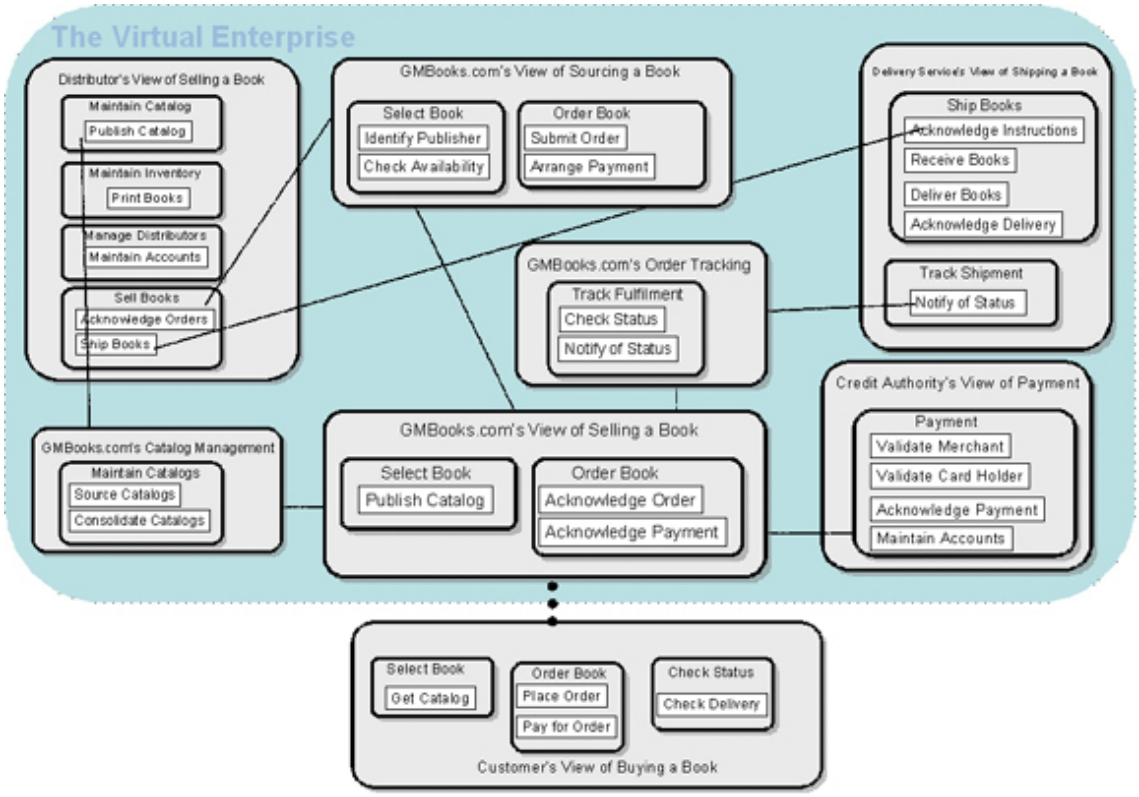


Motivating "Document Engineering"

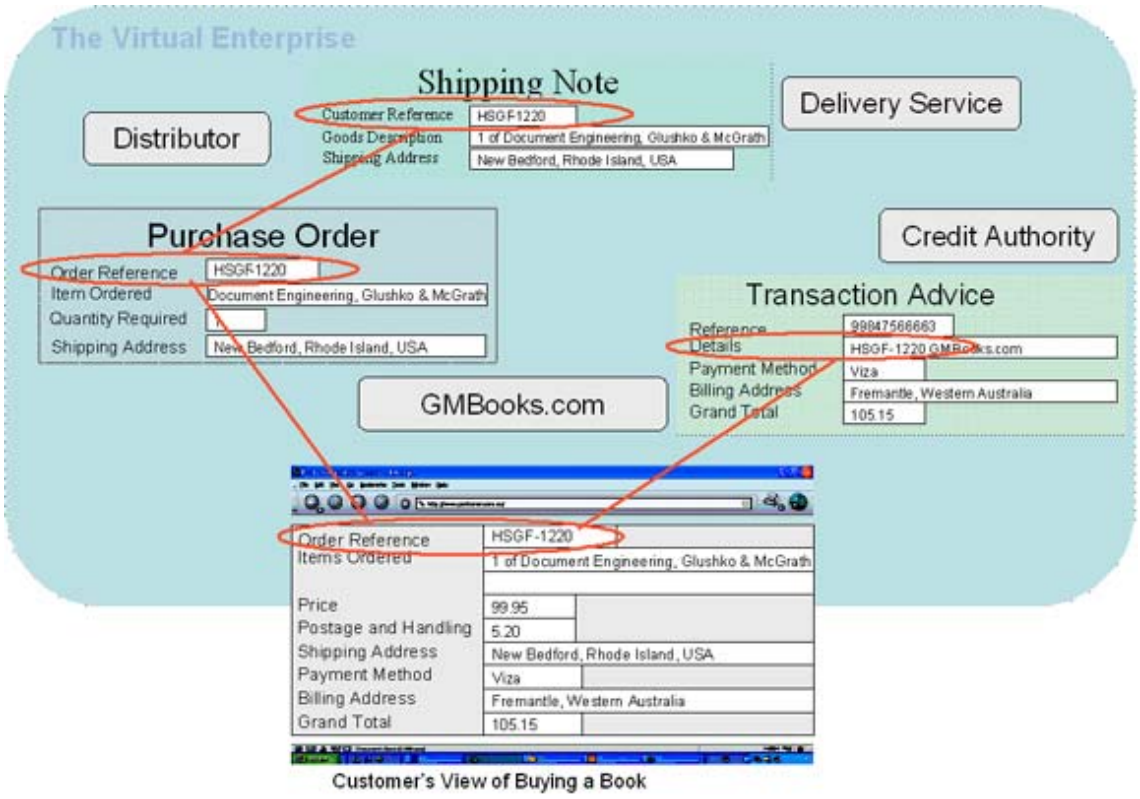
- Scenario:
  - Customer selects book from catalog on an online bookstore
  - Customer pays with credit card
  - Book arrives via express shipper two days later
- From the customer's perspective there is only one "transaction"
- But the bookstore is a virtual enterprise that follows the drop shipment pattern to coordinate the activities of 4 different service providers transacting with each other
- This coordination - or choreography - is carried out with document exchanges

---

The Virtual Bookstore



Overlapping Information Models in the Virtual Bookstore



Customer's View of Buying a Book

An Ancient Business Document

On the 16th of
Tammuz, year 4
of Artaxerces,
Halfat brought
barley: 1 kor,
12 seah, 3 qab;
wheat: 1 kor,
5 seah, 4 qab.

Document Exchange Patterns

- Halfat's clay pot receipt for taxes is certainly one of the oldest documents that record a business transaction (355 BCE)
- Businesses have long dealt with each other by exchanging documents
- We use concepts like "supply chains" and "distribution channels" as metaphors for the coordinated or choreographed flow of information and materials/products between businesses
- These are complex patterns composed from the document exchange pattern

---

The Evolution of "Business Architectures" for Document Exchange

- The technology for business documents has changed throughout history, but the basic idea of document exchange has changed relatively little
- For over two thousand years the "business architecture" around the exchange of documents was non-proprietary and loosely-coupled
- Neither party to the exchange needed to know how the other produced or understood the documents
- A very short time period (evolutionarily speaking) from about 1950-1995 used proprietary and tightly-coupled business architecture

---

Document Engineering as the Methodology for Exploiting the Internet as a Business Application Platform

- But being non-proprietary and loosely-coupled isn't sufficient for a successful document exchange – exchanging information does no good if the information can't be understood by the parties (or applications) doing the exchanging.
- The Web services "standards" not only don't solve this problem – they completely ignore it
- Document Engineering will ensure that the documents can be understood

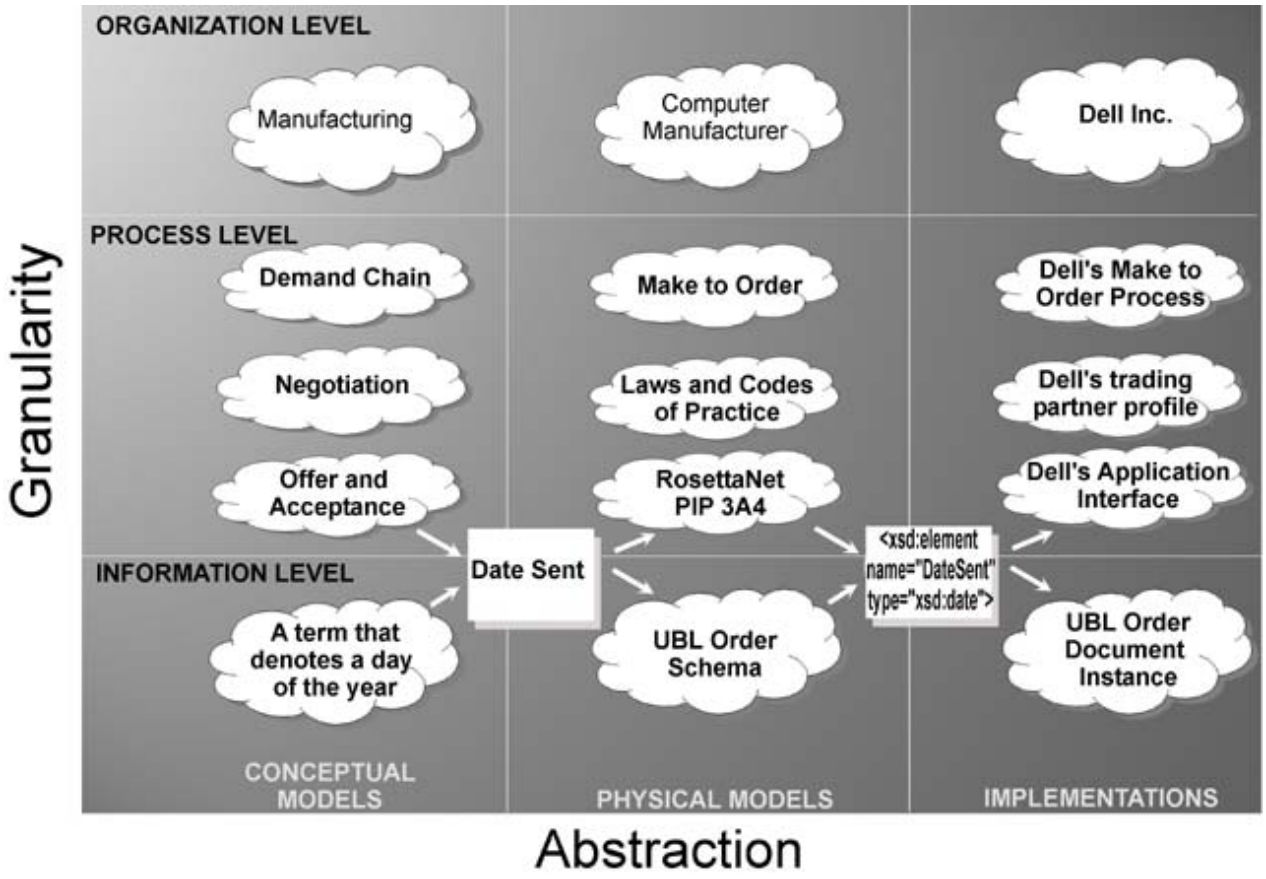
---

Document Exchange is the Mother of All Patterns

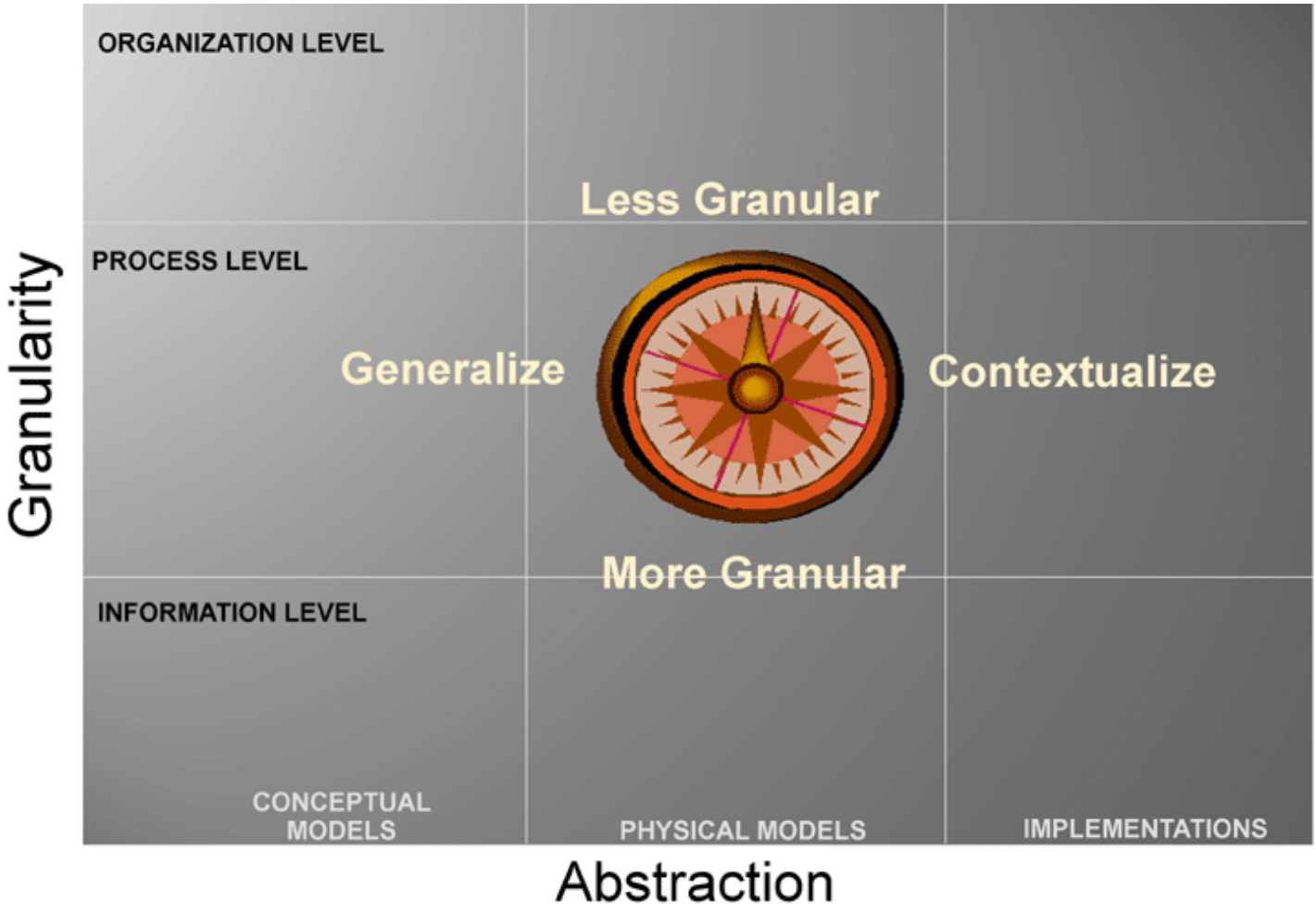
- Document exchange is the "mother of all patterns" for business models, business processes, and business information:
- *Business model or organizational* patterns: marketplace, auction, supply chain, build to order, drop shipment, vendor managed inventory, etc.

- *Business process* patterns: procurement, payment, shipment, reconciliation, etc.
- *Business information* patterns: catalog, purchase order, invoice, etc. and the components they contain for party, time, location, measurement, etc.

The Model Matrix



The "Pattern Compass"



Patterns in Document Engineering

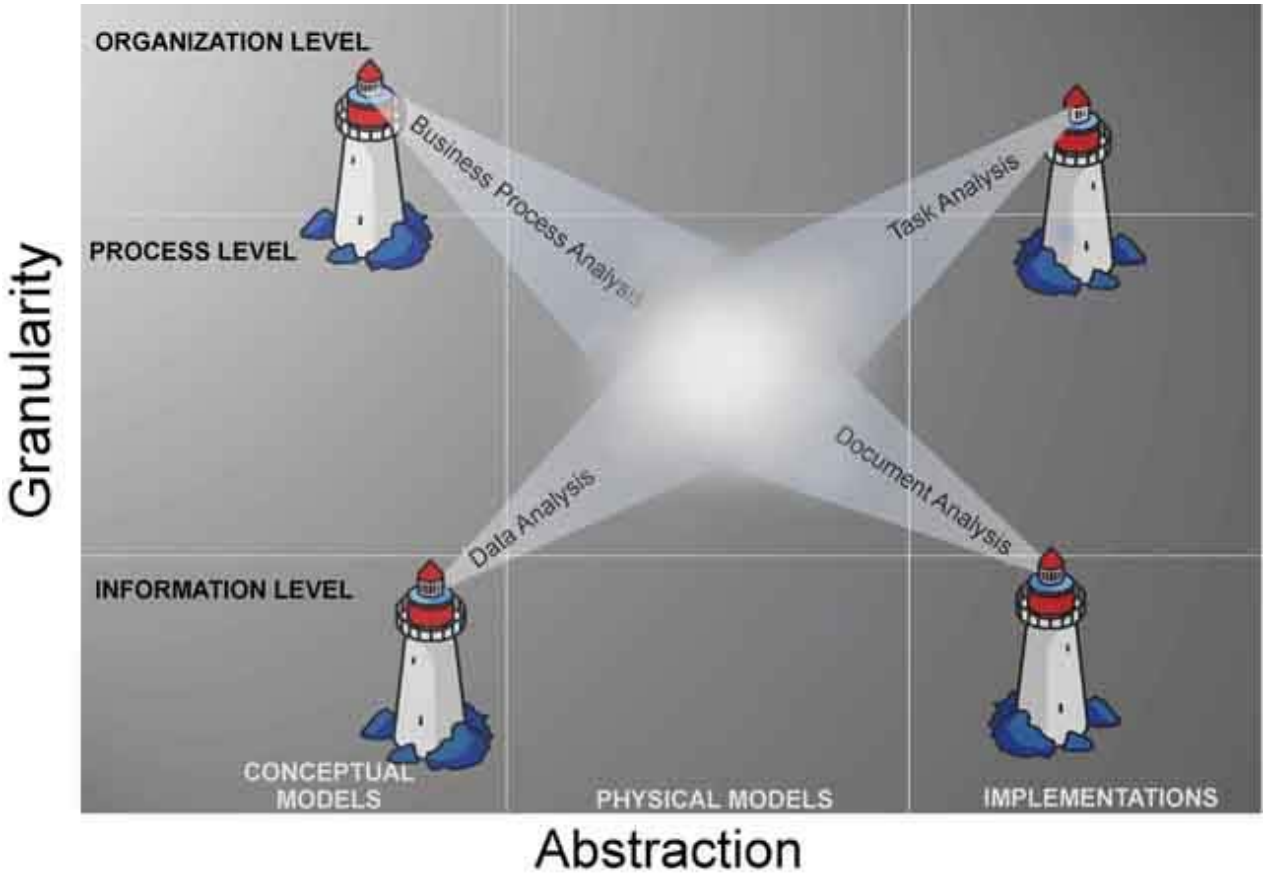
- The essence of Document Engineering is its systematic approach for discovering and exploiting the relationships between patterns of different types
- Working from the top down to ensure that a business model is feasible
- Working from the bottom up to ensure that we are designing and optimizing the activities that add the most value
- We need models of the desired business processes and the documents that they will produce and consume at the same level of detail and implementability

Meeting in the Middle [1]

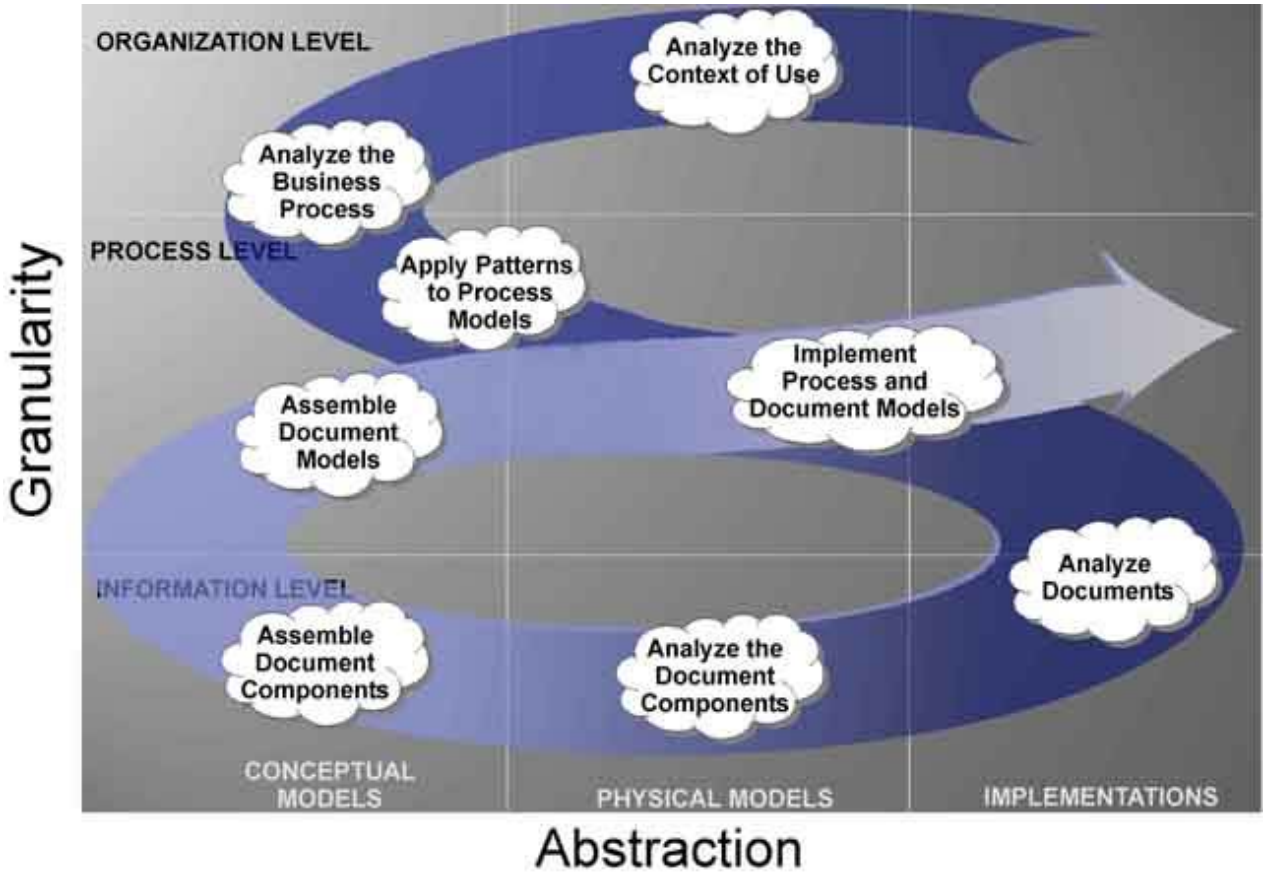
- We need to achieve both business and technical interoperability – the former is necessary but insufficient for the latter
- We need models of the desired business processes and the documents that they will produce and consume at the same level of detail and implementability

- This is represented in the Model Matrix as "meeting in the middle"
- Document Engineering is a systematic approach for "getting to the middle"

Meeting in the Middle [2]



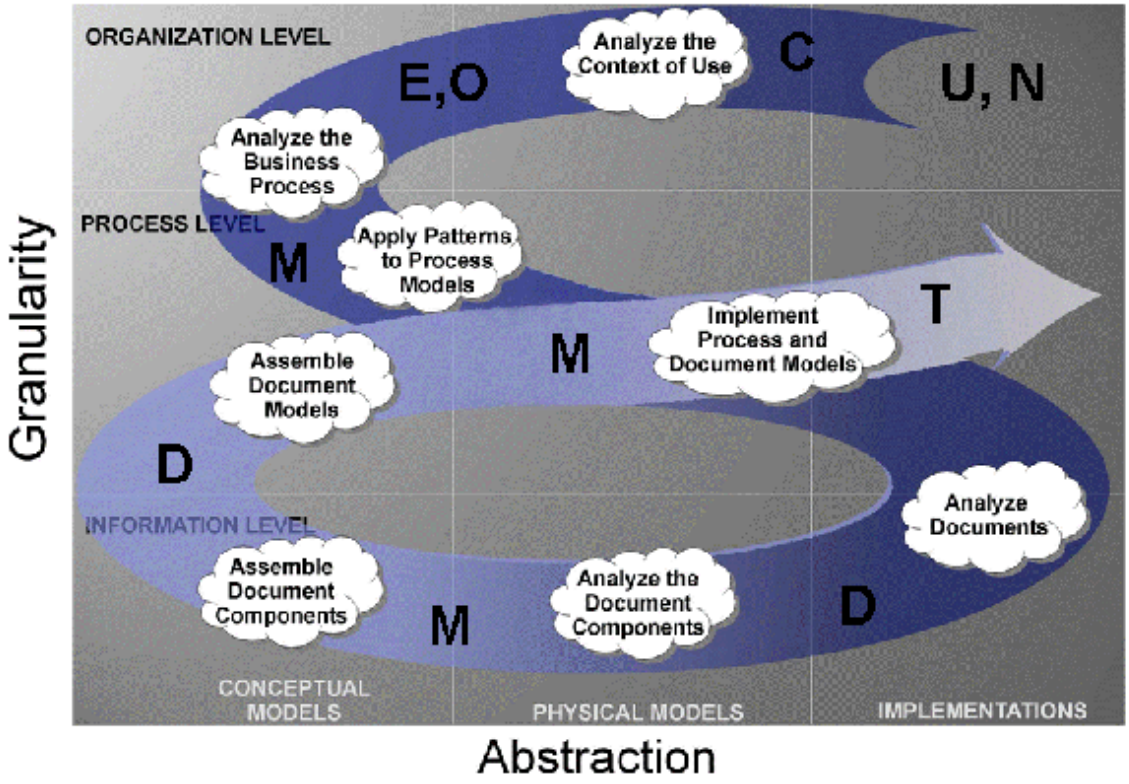
The Document Engineering Approach



A Checklist for Describing Projects and Case Studies

- D -- data types and document types
- O -- organizational processes
- C -- context (types of products or services, industry, geography, regulatory considerations)
- U -- user types and special user requirements
- M -- models, patterns, or standards that apply
- E -- enterprises and eco systems (e.g., trading communities, standards bodies)
- N -- the needs (business case) driving the enterprise(s)
- T -- technology constraints and opportunities

D-O-C-U-M-E-N-T in the Document Engineering Approach

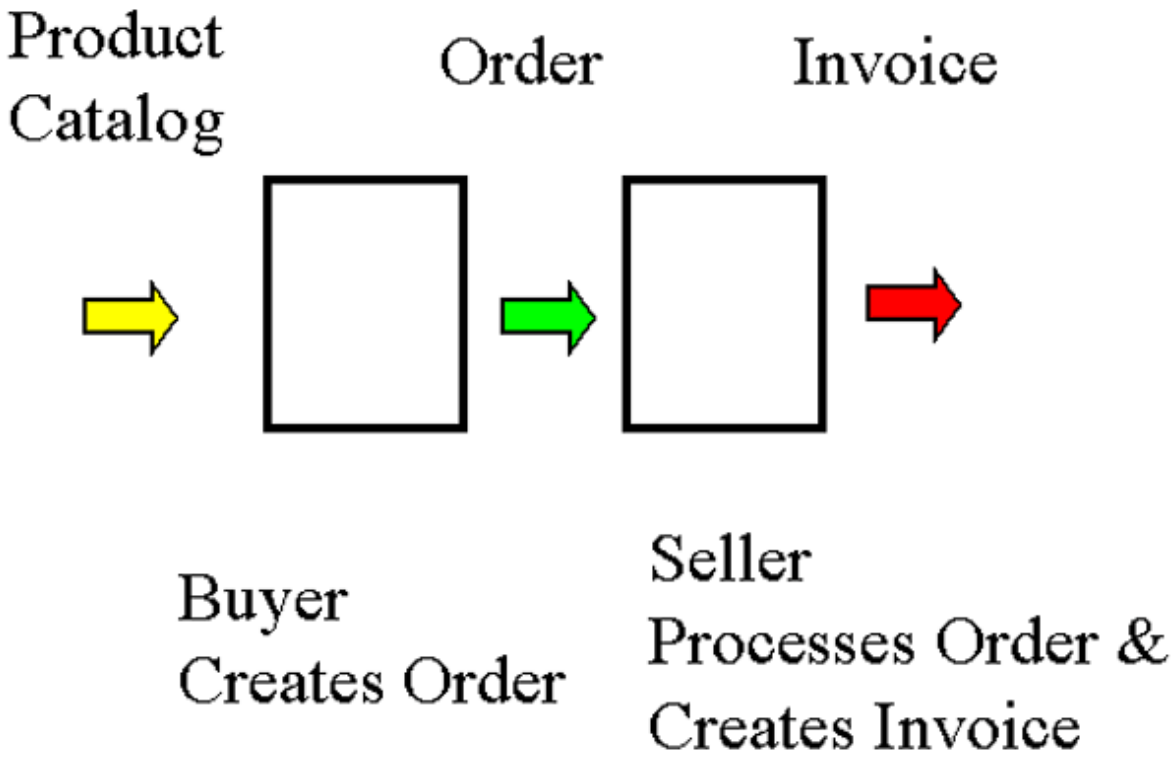


Modeling Documents {and,vs,or} Modeling Processes

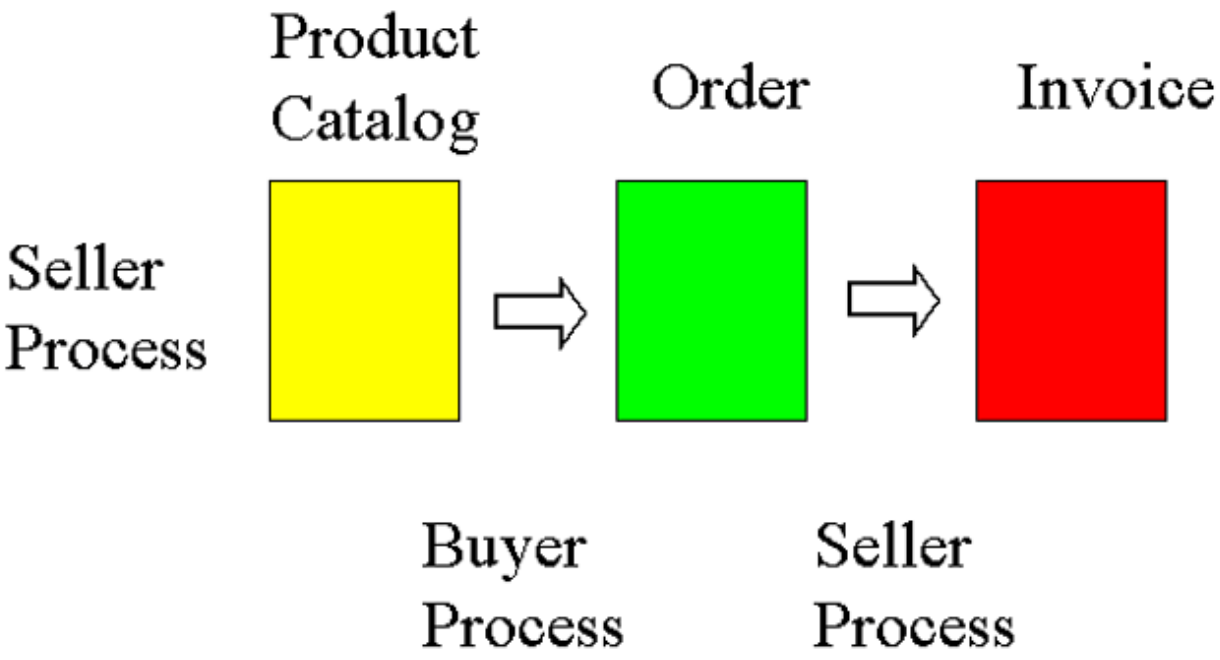
- Documents are always the result of some process and often the input to another one
- This is most evident for transactional documents where patterns of paired document exchange are the building blocks for supply chains, marketplaces, auctions and other business patterns
- By understanding the information in the documents, we learn what kinds of processes are possible
- By understanding the processes, we learn what kinds of information are needed

---

A Process-Centric Depiction



A Document-Centric Depiction



Benefits of a Document-Centric Modeling Approach

- Documents are more tangible than processes, easier to analyze and communicate

- SOA emphasizes documents as the public interfaces to private processes
- David Cohn: 100,000 nouns enable us to understand the meanings of 10,000 verbs

---

- # DOCUMENT ENGINEERING AND XML

Document Engineering Isn't Just About XML

- XML is a useful technology for Document Engineering, but using XML doesn't make you a document engineer
- The *best* thing about XML is the ease with which you can create a new vocabulary for a particular type of document
- XML is just the syntax in which we encode document models... what really matters is how we modeled the documents

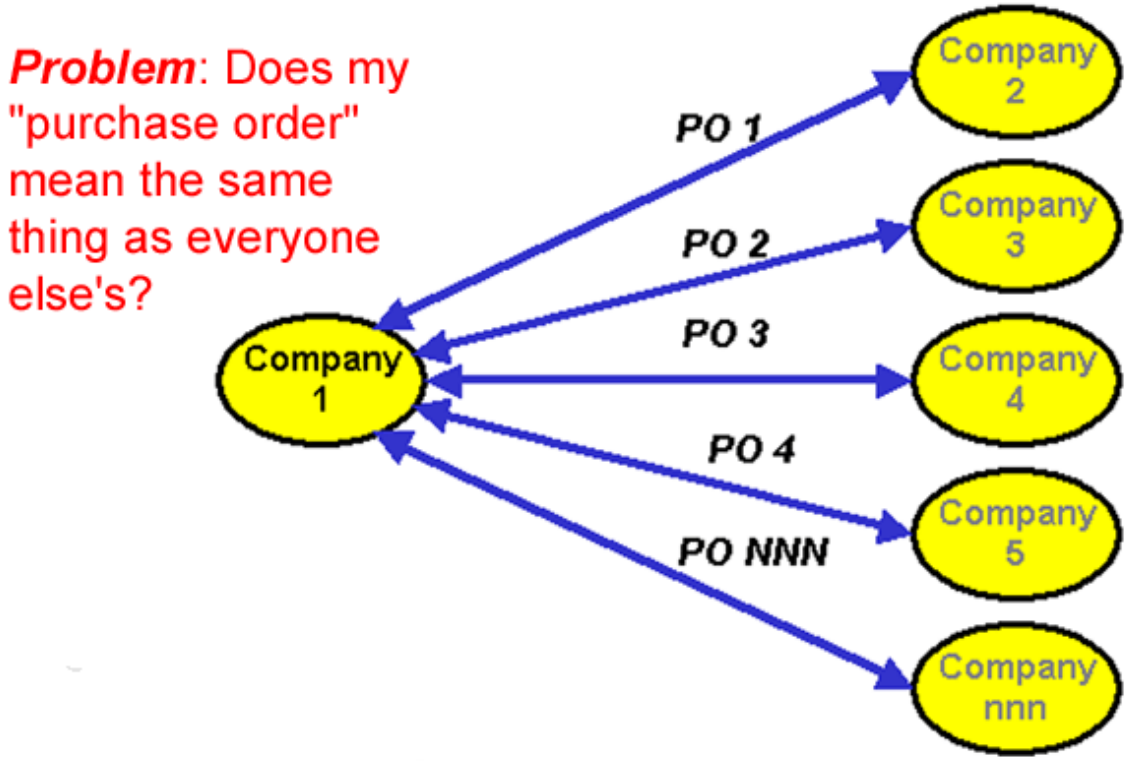
---

Creating Models is Easy, But Creating GOOD Models is Hard

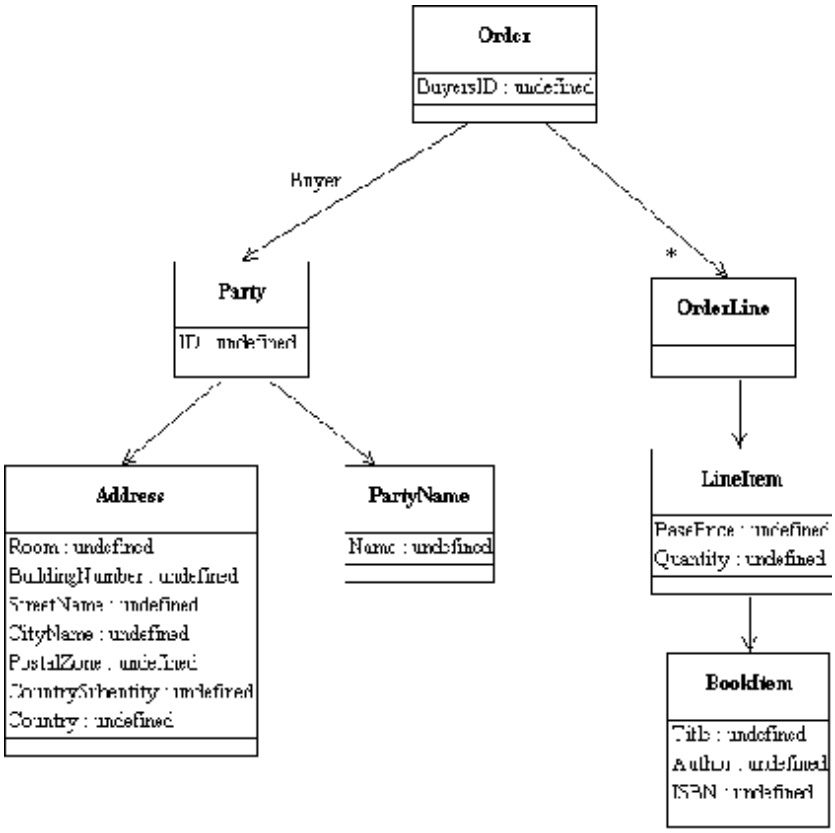
- The *worst* thing about XML is the same as the best thing – the ease with which you can create a new vocabulary
- No way around the classical problems of classification and naming we know from philosophy, linguistics, cognitive psychology, and information science
- XML is NOT "self-describing"
- There are often multiple vocabularies for the same or related domains and especially for the common information models that are used in more than one domain

---

The Equivalence Problem



The Target Model For The Interoperability Scenarios



The XSD Schema for the Expected Order [1]

- ```
  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified">

  <xs:element name="Order" type="OrderType"/>
  <xs:complexType name="OrderType">
    <xs:sequence>
      <xs:element name="BuyersID" type="xs:string"/>
      <xs:element name="BuyerParty" type="PartyType"/>
      <xs:element name="OrderLine"  type="OrderLineType"
            maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="PartyType">
  <xs:sequence>
      <xs:element name="ID" type="xs:string"/>
      <xs:element name="PartyName" type="PartyNameType"/>
      <xs:element name="Address" type="AddressType"/>
  </xs:sequence>
  </xs:complexType>

  <xs:complexType name="PartyNameType">
  <xs:sequence>
      <xs:element name="Name" type="xs:string" minOccurs="0"/>
  </xs:sequence>
  </xs:complexType>
  ```

The XSD Schema for the Expected Order [2]
  ```

```
<xs:complexType name="AddressType">
<xs:sequence>
    <xs:element name="Room" type="xs:string"/>
    <xs:element name="BuildingNumber" type="xs:string"/>
    <xs:element name="StreetName" type="xs:string"/>
    <xs:element name="CityName" type="xs:string"/>
    <xs:element name="PostalZone" type="xs:string"/>
    <xs:element name="CountrySubentity" type="xs:string"/>
    <xs:element name="Country" type="xs:string"/>
</xs:sequence>
</xs:complexType>

<xs:complexType name="OrderLineType">
<xs:sequence>
    <xs:element name="LineItem" type="LineItemType"/>
</xs:sequence>
</xs:complexType>

<xs:complexType name="LineItemType">
<xs:sequence>
    <xs:element name="BookItem" type="BookItemType"/>
    <xs:element name="BasePrice" type="xs:decimal"/>
    <xs:element name="Quantity" type="xs:int"/>
</xs:sequence>
</xs:complexType>

<xs:complexType name="BookItemType">
<xs:sequence>
    <xs:element name="Title" type="xs:string"/>
    <xs:element name="Author" type="xs:string"/>
    <xs:element name="ISBN" type="xs:string"/>
</xs:sequence>
</xs:complexType>
</xs:schema>
```

The Expected Instance

```
<Order>
<BuyersID>91604</BuyersID>
<BuyerParty>
<ID>KEEN</ID>
<PartyName>
   <Name>Maynard James Keenan</Name>
</PartyName>
<Address>
   <Room>505</Room>
   <BuildingNumber>11271</BuildingNumber>
   <StreetName>Ventura Blvd.</StreetName>
   <CityName>Studio City</CityName>
   <PostalZone>91604</PostalZone>
   <CountrySubentity>California</CountrySubentity>
   <Country>USA</Country>
</Address>
</BuyerParty>
<OrderLine>
<LineItem>
   <BookItem>
     <Title>Foucault's Pendulum</Title>
     <Author>Umberto Eco</Author>
     <ISBN>0345368754</ISBN>
     </BookItem>
     <BasePrice>7.99</BasePrice>
     <Quantity>1</Quantity>
</LineItem>
</OrderLine>
```

```
</Order>
```

Identical Model with Different Tag Names [1]

- ```
  <Customer>
  <Number>KEEN</Number>
  <Name>
     <BusinessName>Maynard James Keenan</BusinessName>
  </Name>

  <Location>
     <Unit>505</Unit>
     <StreetNumber>11271</StreetNumber>
     <Street>Ventura Blvd.</Street>
     <City>Studio City</City>
     <ZipCode>91604</ZipCode>
     <State>California</State>
     <Country>USA</Country>
  </Location>
  </Customer>
  ```

Identical Model with Different Tag Names [2]

- ```
  <Acheteur>
  <ID>KEEN</ID>
  <Nom>
     <NomCommercial>Maynard James Keenan</NomCommercial>
  </Nom>
  <Addresse>
     <Appartment>505</Appartment>
     <BÃ¢timent>11271</BÃ¢timent>
     <Rue>Ventura Blvd.</Rue>
     <Ville>Studio City</Ville>
     <CodePostal>91604</CodePostal>
     <Etat>California</Etat>
     <Pays>USA</Pays>
  </Addresse>
  </Acheteur>
  ```

Same Model, Attributes Instead of Elements

- ```
  <BuyerParty
   ID="KEEN"
   Name="Maynard James Keenan"
   Room="505" BuildingNumber="11271"
   StreetName="Ventura Blvd."
   City="Studio City"
   State="California"
   PostalCode="91604"
   >
  ```

Granularity Conflicts
  ```

- <Address>
  <StreetAddress>11271 Ventura Blvd. #505</StreetAddress>
  <City>Studio City 91604</City>
  <CountrySubentity>California</CountrySubentity>
  <Country>USA</Country>
  </Address>

  <PartyName>
  <FamilyName>Keenan</FamilyName>
  <MiddleName>James</MiddleName>
  <FirstName>Maynard</FirstName>
  </PartyName>

Assembly Mismatch - Separate Customer and Order Documents [1]

- <BuyerParty>
  <ID>KEEN</ID>
  <PartyName>
  <Name>Maynard James Keenan</Name>
  </PartyName>
  <Address>
  <Room>505</Room>
  <BuildingNumber>11271</BuildingNumber>
  <StreetName>Ventura Blvd.</StreetName>
  <CityName>Studio City</CityName>
  <PostalZone>91604</PostalZone>
  <CountrySubentity>California</CountrySubentity>
  <Country>USA</Country>
  </Address>
  </BuyerParty>

Assembly Mismatch - Separate Customer and Order Documents [2]

- <Order>
  <BuyersID>91604</BuyersID>
  <BuyerParty>
  <ID>KEEN</ID>
  </BuyerParty>
  <OrderLine>
  <LineItem>
  <BookItem>
  <Title>Foucault's Pendulum</Title>
  <Author>Umberto Eco</Author>
  <ISBN>0345368754</ISBN>
  </BookItem>
  <BasePrice>7.99</BasePrice>
  <Quantity>1</Quantity>
  </LineItem>
  </OrderLine>
  </Order>

Conceptual Incompatibility

- ```
  <Address>
     <Latitude direction="N">37.871</Latitude>
     <Longitude direction="W">-122.271</Longitude>
  </Address>
  ```

Validation Does Not Imply Interoperability

- After all these cases where interoperability may or may not possible because the conceptual or implementation models differ we need to talk about the "easy" case ... and make sure you recognize that it might not be
- Suppose the document validates against the recipient's schema
  - The semantics can still be different in important ways (the ID SSN example) – the strongest level of validation can fall short of establishing that the "same tags" have exactly the "same meaning" to the sender and recipient
  - Furthermore, the recipient may not be able to validate all of the business rules that are important
  - This is a good argument for industry standards / reference models / in your conceptual models or using XML vocabularies that represent them in authoritative ways

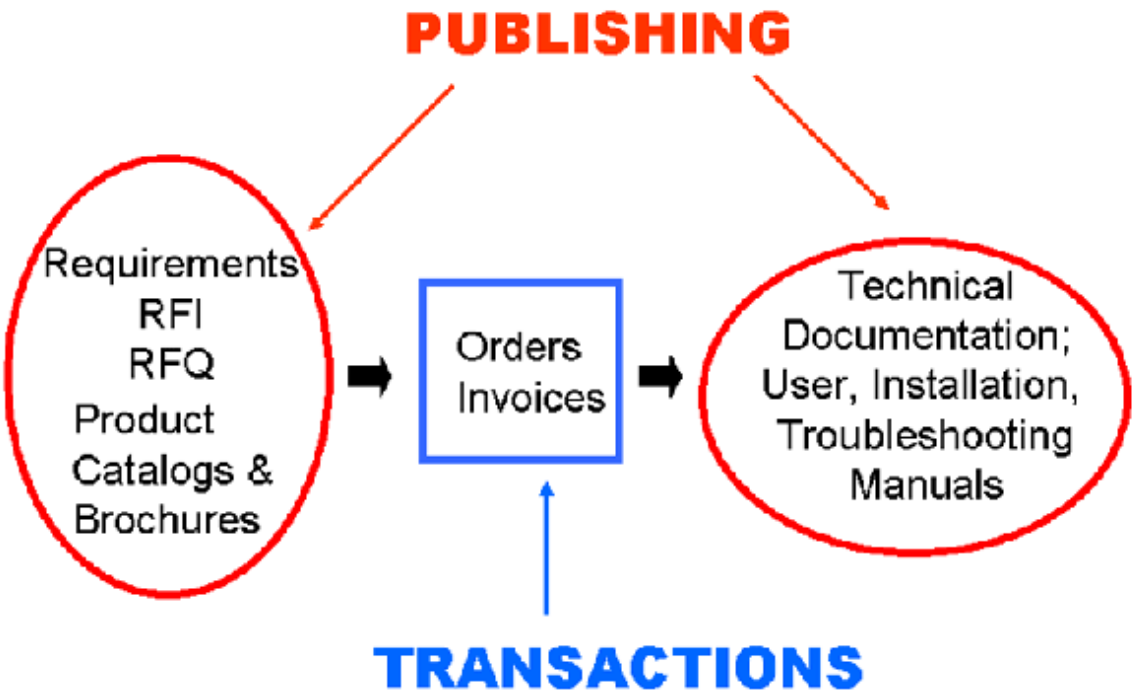
---

METHODS FOR MODELING COMPONENTS

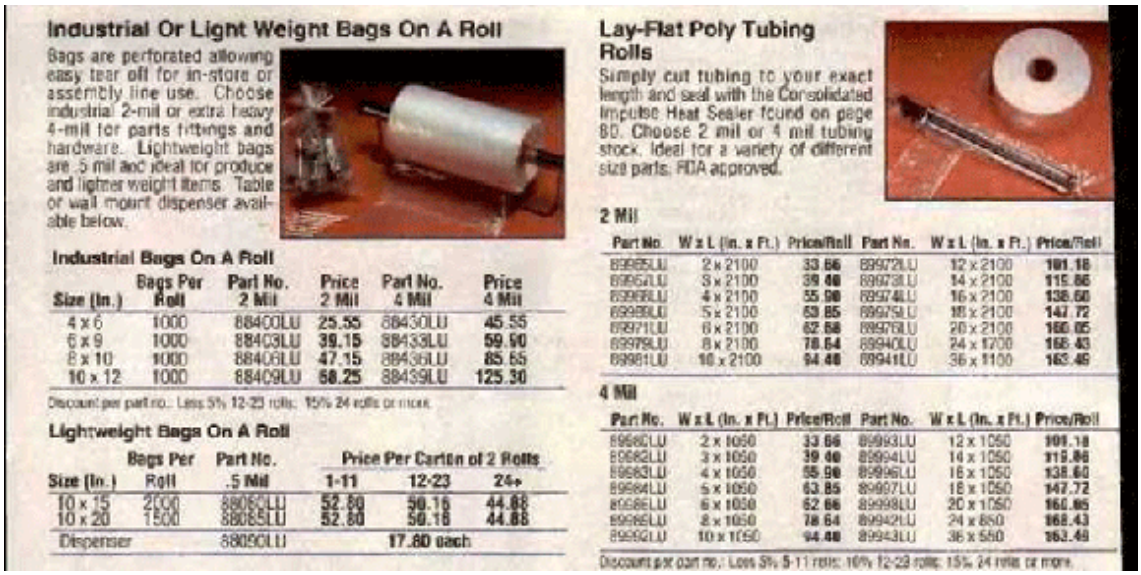
Documents vs. Data

- Many people have contrasted "documents" and "data" and concluded that documents and data cannot be understood and handled with the same terminology, techniques, and tools.
- This document vs. data distinction is embedded and reinforced in XML textbooks, technology, and product marketing
- And it doesn't always help

Mixing Data and Documents
  ```



Data or Document?



Industrial Or Light Weight Bags On A Roll

Bags are perforated allowing easy tear off for in-store or assembly line use. Choose industrial 2-mil or extra heavy 4-mil for parts fittings and hardware. Lightweight bags are .5 mil and ideal for produce and lighter weight items. Table or wall mount dispenser available below.

Industrial Bags On A Roll

Size (In.)	Bags Per Roll	Part No. 2 Mil	Price 2 Mil	Part No. 4 Mil	Price 4 Mil
4 x 6	1000	88400LU	25.55	88430LU	45.55
6 x 9	1000	88403LU	39.15	88433LU	59.90
8 x 10	1000	88406LU	47.15	88436LU	85.55
10 x 12	1000	88409LU	58.25	88439LU	125.30

Discount per part no.: Less 5% 12-23 rolls; 15% 24 rolls or more.

Lightweight Bags On A Roll

Size (In.)	Bags Per Roll	Part No. .5 Mil	Price Per Carton of 2 Rolls 1-11	12-23	24+
10 x 15	2000	88060LU	52.80	50.16	44.88
10 x 20	1500	88065LU	52.80	50.16	44.88
Dispenser		88090LU		17.80 each	

Lay-Flat Poly Tubing Rolls

Simply cut tubing to your exact length and seal with the Consolidated Impulse Heat Sealer found on page 80. Choose 2 mil or 4 mil tubing stock. Ideal for a variety of different size parts. FDA approved.

2 Mil

Part No.	W x L (In. x Ft.)	Price/Roll	Part No.	W x L (In. x Ft.)	Price/Roll
89965LU	2 x 2100	33.66	89972LU	12 x 2100	101.18
89964LU	3 x 2100	39.48	89973LU	14 x 2100	115.86
89966LU	4 x 2100	55.98	89974LU	16 x 2100	138.60
89969LU	5 x 2100	63.85	89975LU	18 x 2100	147.72
89971LU	6 x 2100	62.86	89976LU	20 x 2100	160.05
89979LU	8 x 2100	78.64	89940LU	24 x 1200	168.43
89981LU	10 x 2100	94.48	89941LU	36 x 1100	163.49

4 Mil

Part No.	W x L (In. x Ft.)	Price/Roll	Part No.	W x L (In. x Ft.)	Price/Roll
89960LU	2 x 1050	33.66	89993LU	12 x 1050	101.18
89962LU	3 x 1050	39.48	89994LU	14 x 1050	115.86
89963LU	4 x 1050	55.98	89996LU	16 x 1050	138.60
89964LU	5 x 1050	63.85	89997LU	18 x 1050	147.72
89986LU	6 x 1050	62.86	89998LU	20 x 1050	160.05
89985LU	8 x 1050	78.64	89942LU	24 x 850	168.43
89992LU	10 x 1050	94.48	89943LU	36 x 550	163.49

Discount per part no.: Less 5% 5-11 rolls; 10% 12-23 rolls; 15% 24 rolls or more.

The Document Type Spectrum

Crossing the Chasm with Document Engineering

- Document Engineering harmonizes the terminology and emphasizes what they have in common rather than highlighting their differences:
- Identifying the presentational, content, and structural components
- Eliminating synonymy and homophony
- Identifying and organizing the "good" content components
- Assembling hierarchical document models to organize components to meet requirements for a specific context for information exchange

---

Three Types of Information In Documents

- We need a vocabulary to classify different kinds of information that we find in documents and sets of data
  - *Content* – "what does it mean" information
  - *Structure* – "where is it" or "how it is organized or assembled" information
  - *Presentation* – "how does it look" or "how is it displayed" information
- The amount and relationships among these three kinds of information varies in different kinds of documents

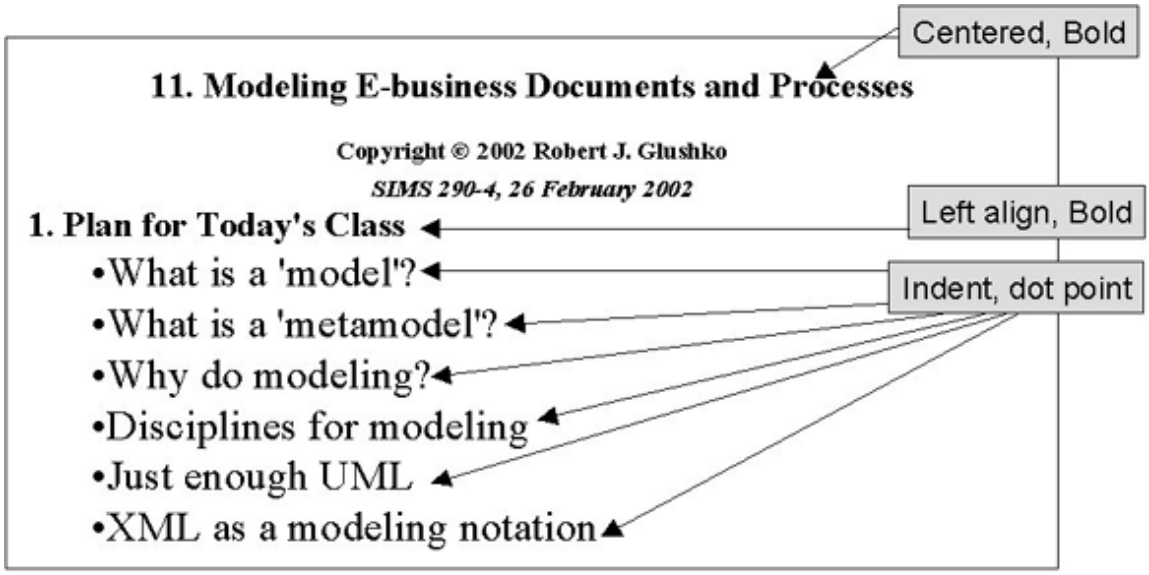
---

Presentation Information

- Human-oriented attributes for visual (or other sensory) differentiation (type font, type size, color, background, indentation, pitch, ...)
- Good user interface design correlates this with structural or content information

- May be concealing structural or content information

Presentation View of a Lecture Slide



Analyzing Presentation Components

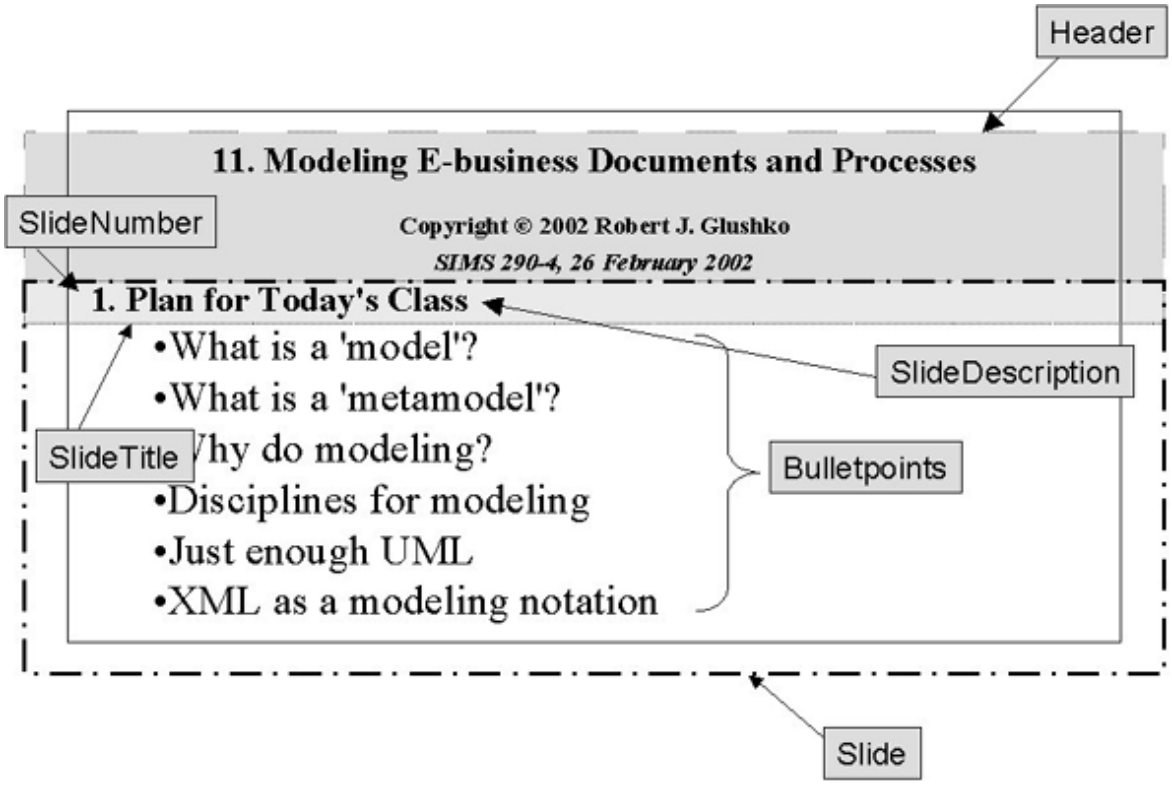
- Presentation affects structure and content by applying transformation rules to them
- To understand the structure and content we must identify and record these rules
- Some transform rules are explicit
- Some transform rules are implicit or ambiguous or misleading

---

Structural Information

- Physical piece of a document (e.g. table, section, title, header, footer)
- Frequently a close relationship between structural and presentation items, especially in a paper document. This goes some way to explaining why the document-centric school places such strong emphasis on structural components.
- Embody the rules on how content components fit together, often hierarchical
- Often driven by context of document use (e.g. overseas address vs. local address)

Structural View of a Lecture Slide



Analyzing Structural Components

- The structural components provide the hierarchical "skeleton" or "scaffold" into which the content components are arranged
- Structural components are often identified by the names attached to pieces of information – think of the outline or table of contents or lists of various kinds
- Metadata to capture
    - Depth of hierarchy
    - Sub-structures included within a structural container
    - Rules for applying numbers or names to content in the hierarchy

---

Content Components

- Content components are the "nouns" in our documents or sets of data – things like "topic," "summary," "name," "address," "price"
- In publications a lot of the content isn't easily identified by "component type" – it may be "just text" that could be playing any of a very large number of roles in the document

- And sometimes you get no help from the set of style or formatting tags in word processors or in HTML, which are very format or structure oriented and not content oriented at all
- We need XML so we can invent the vocabulary of tags needed to describe component content in a specific document type

---

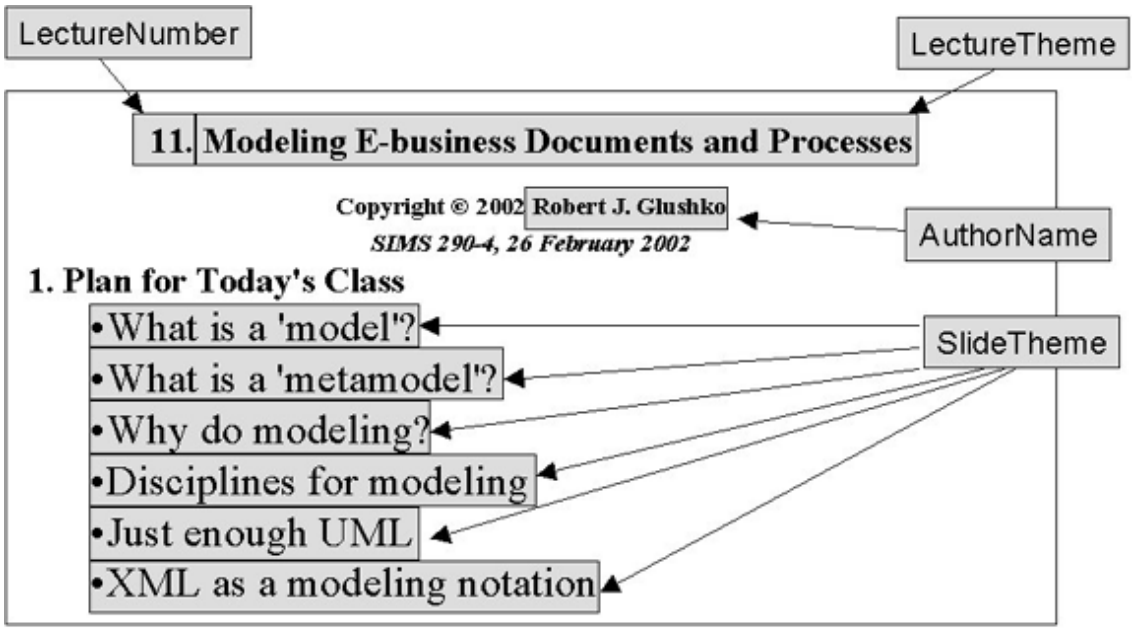
Relationships Among Content Components

- Content components can be related to one another
  - Derivational relationships
  - Referential relationships

Analyzing Content Components

- What attributes about each type of content should we record in our analysis?
  - Names/synonyms/homonyms (what it is called)
  - Definition (what it "means")
  - Roles (what it does)
  - Cardinality/Optionality (occurrence rules)
  - Restricted values, code sets, defaults
  - Data Type (text, numbers, date, video)
  - Relationships/Associations
  - Origin (Is this new information, or from some other source? Who maintains it?)
  - Access (who is allowed to view/change/copy/etc. it)
  - Context (is this information "general purpose" or "core" or is its use limited to specific contexts?)
  - Permanence (is it static or dynamic? how often does it change?)

Content View of a Lecture Slide



Harvesting and Consolidation

- *Harvesting* – Create a set of candidate content components by extracting them from the information sources while removing presentation and structure
- *Consolidation*– Identify synonymns and homonyms among the candidate content components, assigning a unique name to each unique meaning as part of a controlled vocabulary

---

Table of Content Components

Information Elements

Analyzed Documents or Applications

	1	2	3	4	5	6	7	…
1	x	x	x	x	x	x	x	…
2	x	x	x			x		…
3	x	x		x	x		x	…
4	x		x	x				…
5	x	x	x	x	x	x	x	…
6		x		x				…
7	x		x				x	…
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	

Consolidating The Harvest

- We can begin our consolidation with the candidate components from any of the information sources, but we recommend using the one you believe is the most authoritative or that yielded the most components
- The goal is to combine components that are synonyms (different names for the same meaning) and to distinguish any homonyms (same names for different meanings)
- It is desirable for a set of components to enable one and only one way to describe something because duplication or redundancy implies choices that could lead to inconsistent models and non-interoperable schemas

Example Consolidation Table -- Courses

Data Element	Example	COURSE	Online Catalog	Schedule of Classes	Summer Catalog and Schedule	IAS
Course Name		X	X	X	X	X
Course Number	101C	X	X	X	X	X
Semester				X	X	X
CCN				X	X	X
Catalog Description		X	X		X	X

Example Consolidation Table -- Event Calendars

CONSOLIDATED TABLE OF CONTENT COMPONENTS

Name	Semantic Description	Source 1	Source 2	Source 3
Title	The title of the event	X	X	
Start Date	The date of the event, or the first date of a recurring event	X		
End Date	The last date of the event	X		
Location	The location of the event	X	X (merged with synonym *Venue*)	X
Speaker	Name(s) of the person(s) speaking at the event		X	X
Description	The description of the event		X	X
Speaker Title	The title of the speaker			X (renamed homonym *Title*)

Figure 12-13. Completed Consolidated Table of Content Components

Motivating Aggregate Components

- *Atomic* components that hold individual pieces of information
  - Especially in transactional documents,where atomic components have a natural representation as primitive data types ("string," "Boolean," "date") or as datatypes that are derived from these by restriction
- *Document* components that assemble smaller components into the set of information needed to carry out a self-contained purposeful activity
  - Especially in transactional contexts, where documents have a natural correspondence to some unit of work that initiates, records, or responds to a clearly-defined event

Motivating Aggregate Components [2]

- *Aggregate* components are composed of atomic ones and are reused in the assembly of document components
- They are easier to identify in transactional contexts because they are often the key information that flows from one document to another
- "Address" or "Person" are obvious examples of aggregates composed of smaller ones
- Two key questions:
  - How do we select and group atomic components into aggregates?
  - How many aggregates should we create?

Identifying Two Kinds of Component Aggregates

- Structural Aggregates -- sets of components defined by parent-child or containment relationships
- Conceptual Aggregates -- sets of components that "go together" because of logical dependency

---

Identifying Aggregate Components in Non-Transactional Documents [1]

- Aggregates are more elusive on the narrative end of the DTS because there are limits to the rigor with which components can be grouped
- "Mixed content" models arise when there are few or weak constraints on where atomic components can appear
- Presentation often masks the atomic components in potential aggregates
- Structures are often based on conventions for organization and presentation than on semantic relationships
- But there will still generally be components that "go together" to form reusable structures
- And "going together" means different things for each set of components

---

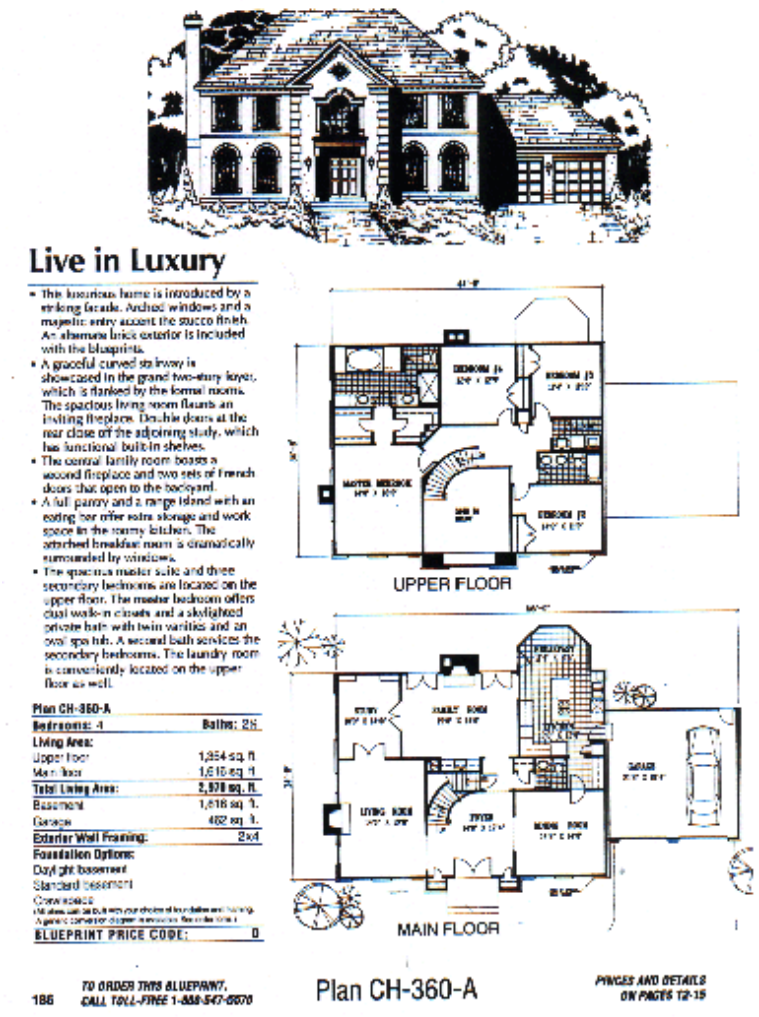
Identifying Aggregate Components in Non-Transactional Documents [2]

- Aggregates can be created in two "bottom-up" ways that focus on the atomic components:
- The first is by rebuilding or making explicit the structures that we took apart in document analysis

- The second is by creating structure in "blobs" of poorly structured information written in an overly narrative style (with mixed content at best)
- A more modular style for the information will increase its regularity and reusability; it will eliminate content that has little value to users and reinforce its use as "boilerplate" or via links

---

Home Plan



Live in Luxury

- This luxurious home is introduced by a striking facade. Arched windows and a majestic entry accent the stucco finish. An alternate brick exterior is included with the blueprints.
- A graceful curved stairway is showcased in the grand two-story foyer, which is flanked by the formal rooms. The spacious living room flaunts an inviting fireplace. Double doors at the rear close off the adjoining study, which has functional built-in shelves.
- The central family room boasts a second fireplace and two sets of French doors that open to the backyard.
- A full pantry and a range island with an eating bar offer extra storage and work space in the sunny kitchen. The attached breakfast room is dramatically surrounded by windows.
- The spacious master suite and three secondary bedrooms are located on the upper floor. The master bedroom offers dual walk-in closets and a skylighted private bath with twin vanities and an oval spa tub. A second bath services the secondary bedrooms. The laundry room is conveniently located on the upper floor as well.

Plan CH-360-A	
Bedrooms: 4	Baths: 2½
Living Area:	
Upper floor	1,354 sq. ft.
Main floor	1,616 sq. ft.
Total Living Area:	2,970 sq. ft.
Basement	1,616 sq. ft.
Garage	462 sq. ft.
Exterior Wall Framing:	2x4
Foundation Options:	
Daylight basement	
Standard basement	
BLUEPRINT PRICE CODE:	D

UPPER FLOOR

MAIN FLOOR

186 TO ORDER THIS BLUEPRINT, CALL TOLL-FREE 1-800-547-6370

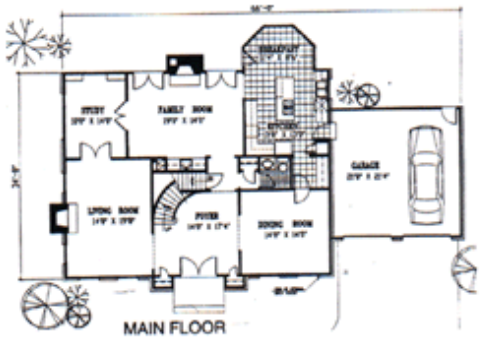
Plan CH-360-A

PRICES AND DETAILS ON PAGES 12-15

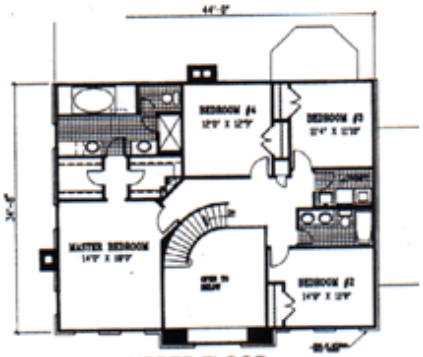
Home Plan "Normalization"

Live in Luxury

- This luxurious home is introduced by a striking facade. Arched windows and a majestic entry accent the stucco finish. An alternate brick exterior is included with the blueprints.



MAIN FLOOR



UPPER FLOOR

- A graceful curved stairway is showcased in the grand two-story foyer, which is flanked by the formal rooms. The spacious living room flaunts an inviting fireplace. Double doors at the rear close off the adjoining study, which has functional built-in shelves.
- The central family room boasts a second fireplace and two sets of French doors that open to the backyard.
- A full pantry and a range island with an eating bar offer extra storage and work space in the roomy kitchen. The attached breakfast room is dramatically surrounded by windows.

- The spacious master suite and three secondary bedrooms are located on the upper floor. The master bedroom offers dual walk-in closets and a skylighted private bath with twin vanities and an oval spa tub. A second bath services the secondary bedrooms. The laundry room is conveniently located on the upper floor as well.

Plan CH-360-A

Plan CH-360-A	
Bedrooms: 4	**Baths:** 2½
Living Area:	
Upper floor	1,354 sq. ft.
Main floor	1,616 sq. ft.
Total Living Area:	**2,970 sq. ft.**
Basement	1,616 sq. ft.
Garage	462 sq. ft.
Exterior Wall Framing:	2x4
Foundation Options:	
Daylight basement	
Standard basement	
Crawlspace	
(All plans can be built with your choice of foundation and framing. A generic conversion diagram is available. See order form.)	
BLUEPRINT PRICE CODE:	D

TO ORDER THIS BLUEPRINT, CALL TOLL-FREE 1-800-547-5570

PRICES AND DETAILS ON PAGES 12-15

Normalization

- *Normalization*– Applying techniques for reducing redundancy and increasing integrity in information

models

- The consolidated list of unique candidate components is equivalent to 1NF in relational theory
- Data normalization techniques can be applied to further refine the set of candidate components (if used sensibly)
- Components that are functionally independent of each other are separated and their bi-directional relationships are recorded

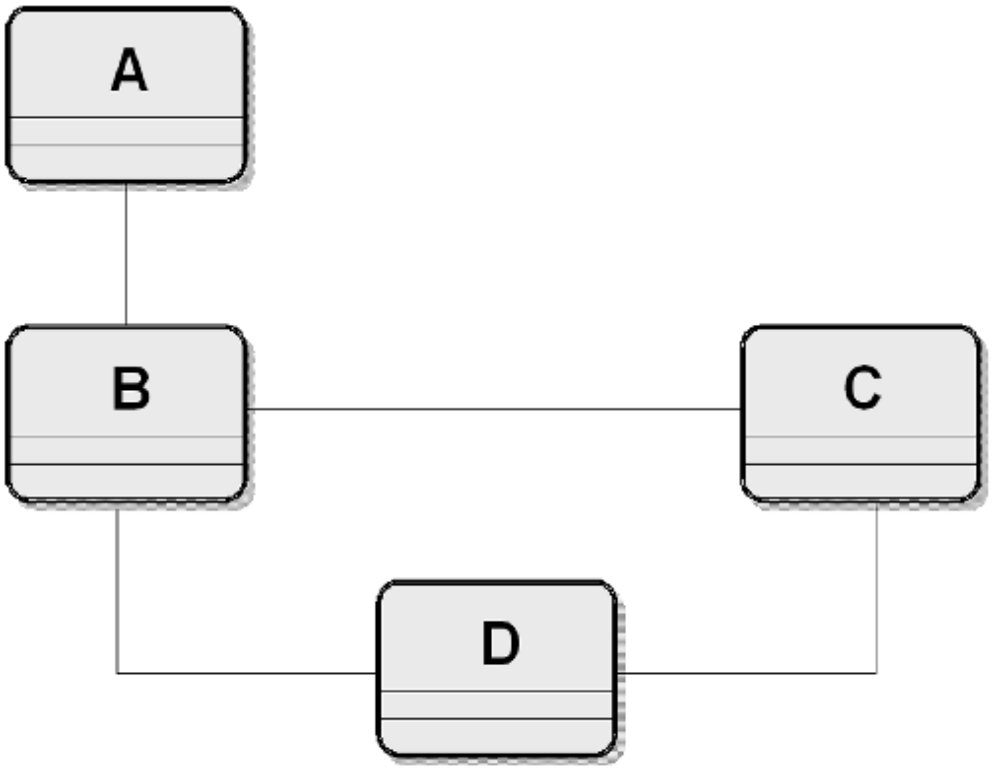
---

The Component Model

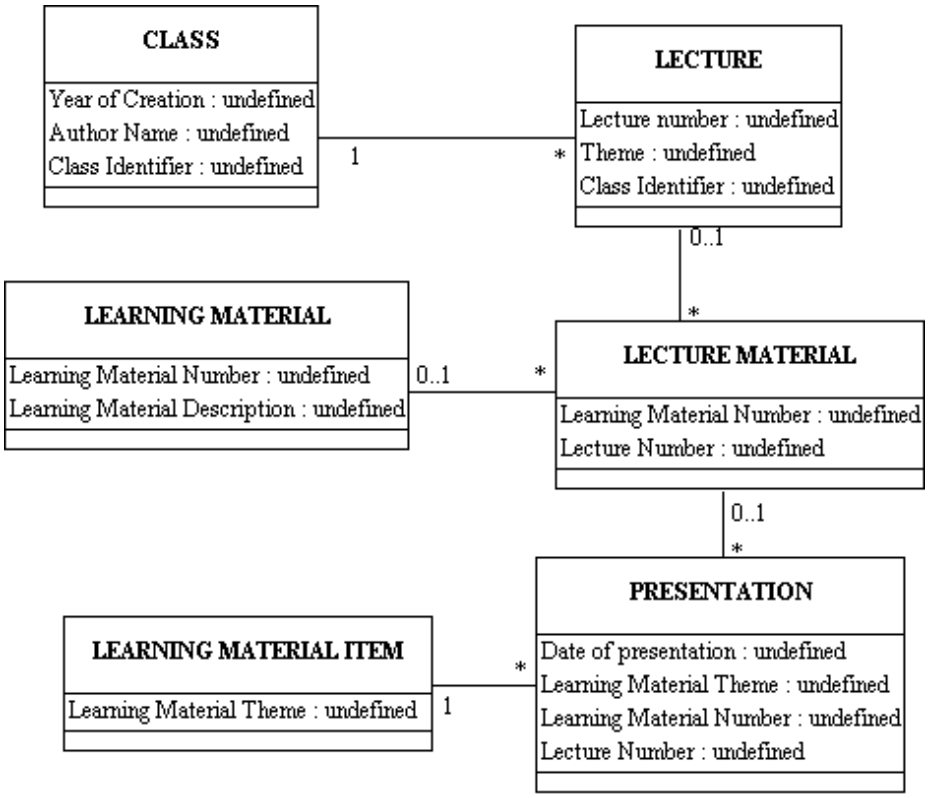
- We have identified primitive and aggregate components
- We've used heuristic or formal means (or both)
- The methods we used and the results reflect the mixture of transactional and non-transactional documents in our context
  - Number of components
  - Size of components
  - Precision of rules for datatypes, associations, cardinality

---

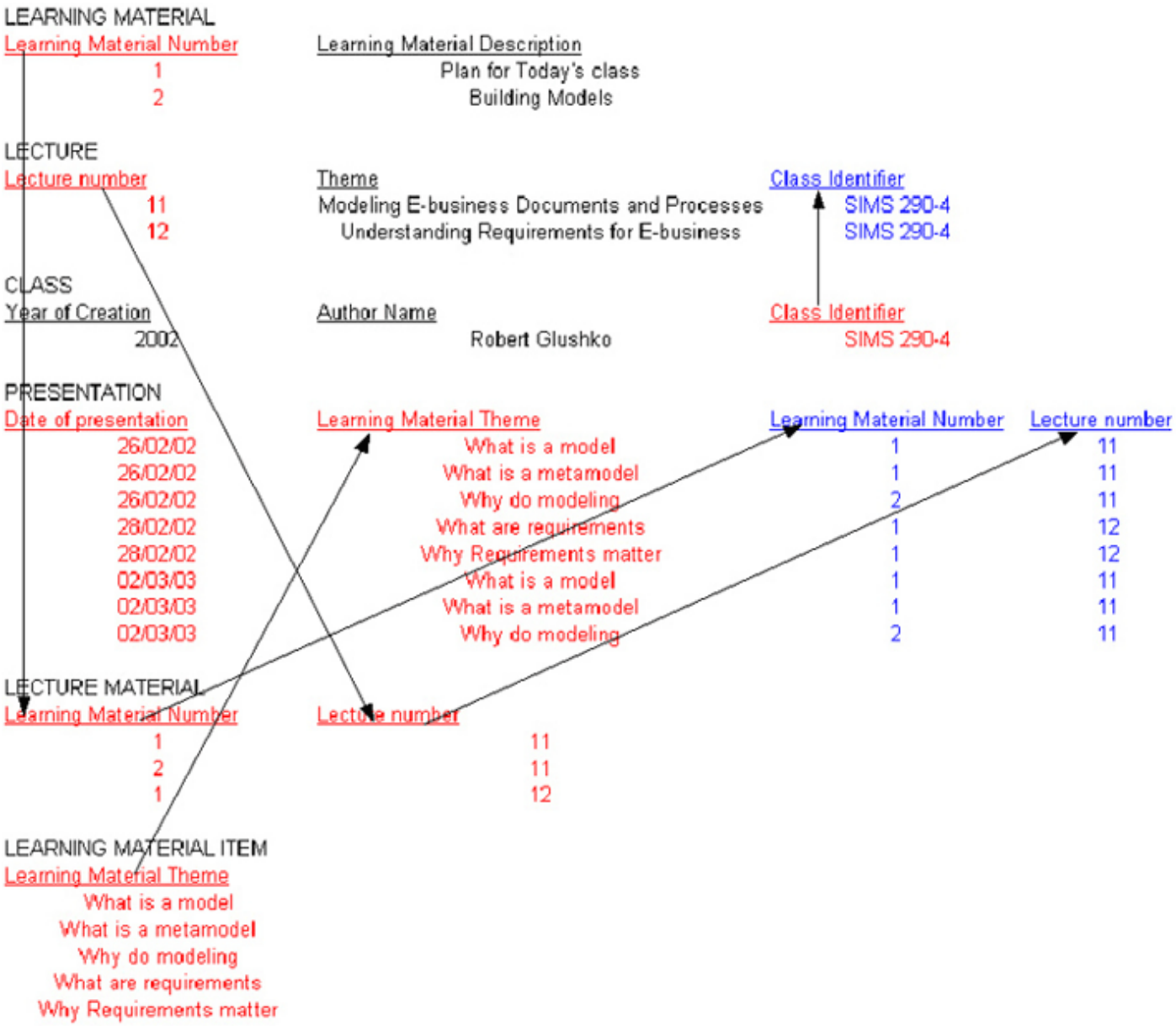
The Component Model is a Set of Relations



Representations of Normalized Models - UML Class Diagram



Representations of Normalized Model - Primary Key Path



- **METHODS FOR MODELING DOCUMENTS**

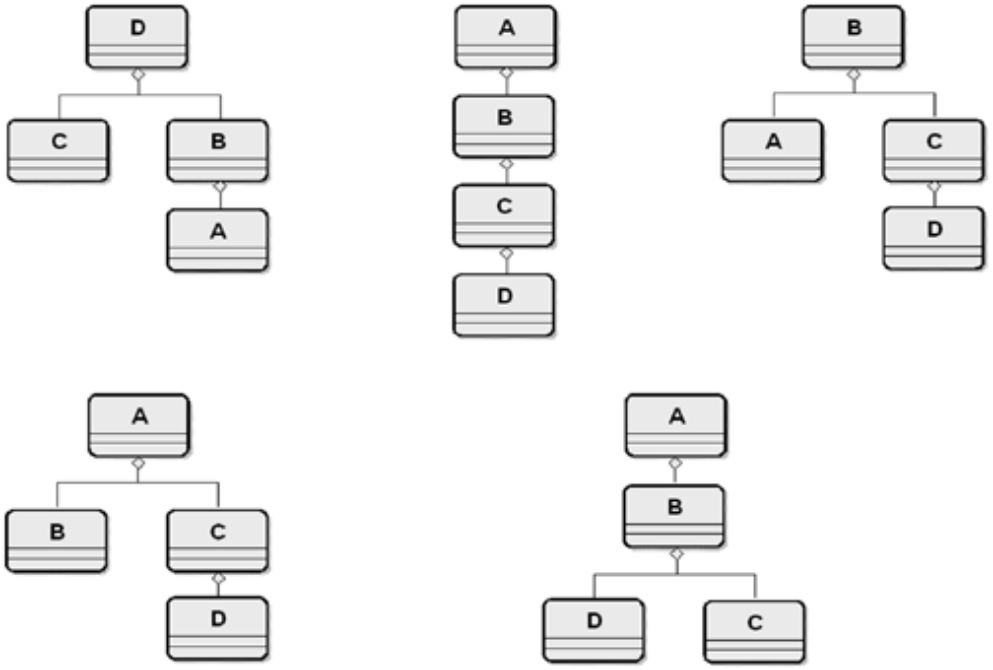
Why We Need Hierarchical Document Models

- A relational model simultaneously describes all of the associations among the components; put another way, it doesn't highlight any particular association
- But when we exchange information, we do so to satisfy the requirements in some context
- If there are multiple ways to interpret the content we will not achieve interoperability
- Hierarchies (tree structures) provide unambiguous structures

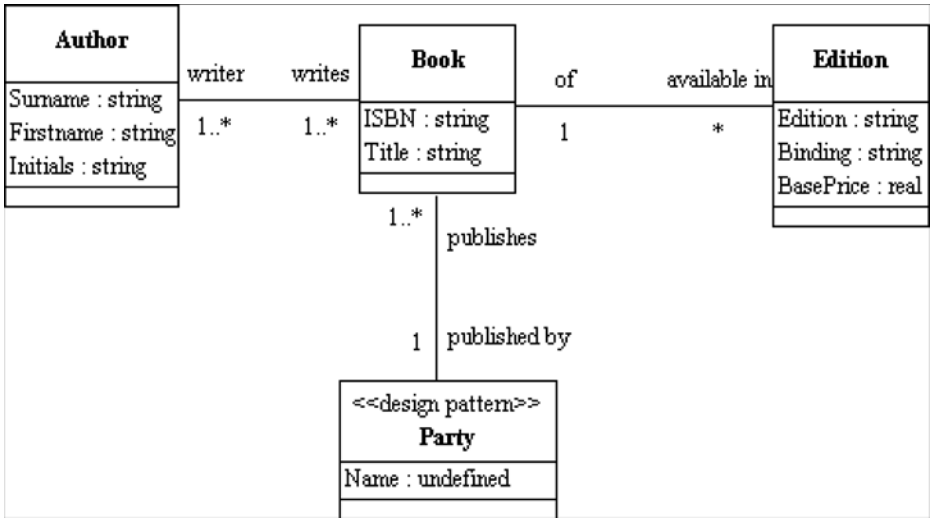
- So we impose a contextual interpretation when we create a hierarchy on a relational model

---

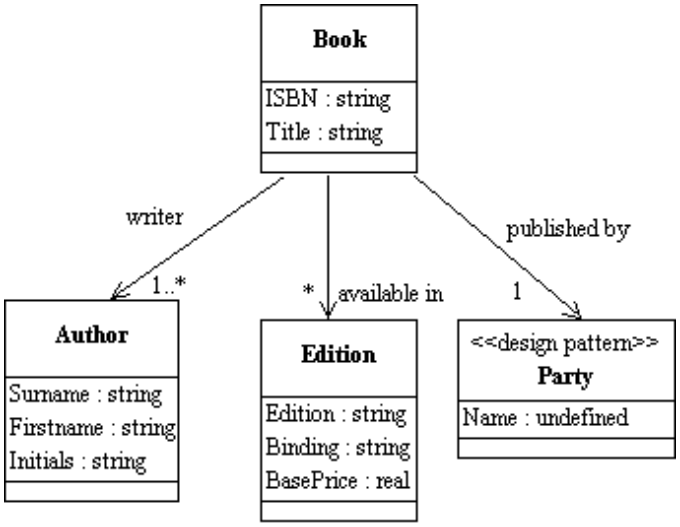
Multiple Paths Through the Component Network



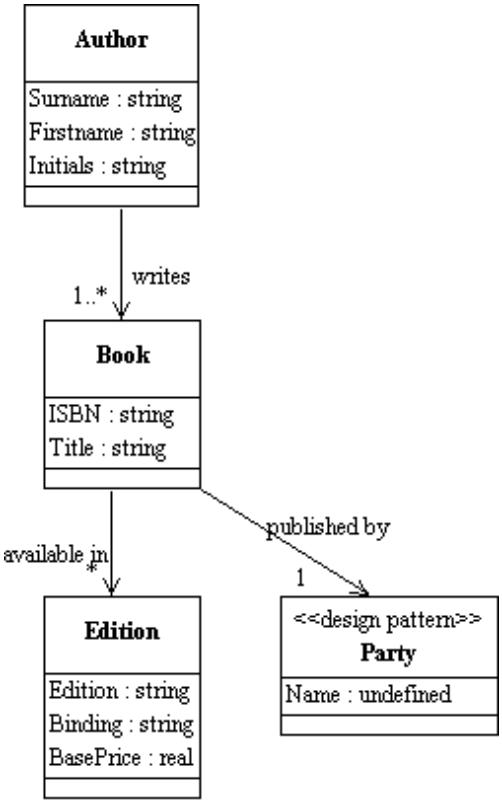
Simple Example -- Book / Author / Edition / Publisher



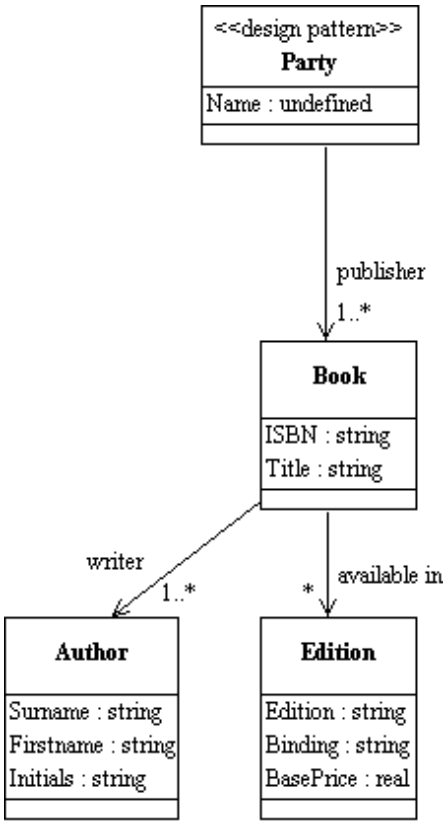
Hierarchical Interpretation of the "Book" Model [1]



Hierarchical Interpretation of the "Book" Model [2]



Hierarchical Interpretation of the "Book" Model [3]



Document Model Assembly

- Document model assembly is the process of creating a model of a document type – hierarchical and nested – by drawing on the "pool" or library of content and structural components
- Assembly involves designing (or selecting a pattern for) the top level structure as an entry point and then navigating through the relationships in the conceptual model collecting the components in the order that best satisfies your requirements
- Assembly order can differ whenever there is a bi-directional relationship between components – whenever two components are functionally independent, an assembly order chooses one of the relationships to enforce an interpretation on the assembled document
- The direction of following the relationship determines which of the structural roles is being used
- We end up with a specific context-sensitive view of the model
- This is the logical basis of the document schema – all we have left to do is to encode it as an XML schema

---

Document Model Assembly and the Document Type Spectrum

- The basic problem of assembly is the same for all types of documents but the solution is different at

different points on the spectrum

- Non-transactional / narrative / publication type documents usually have fewer content-based rules, but their assembly is often shaped by structural or presentation rules

---

Document Model Assembly - Transactional Document Types

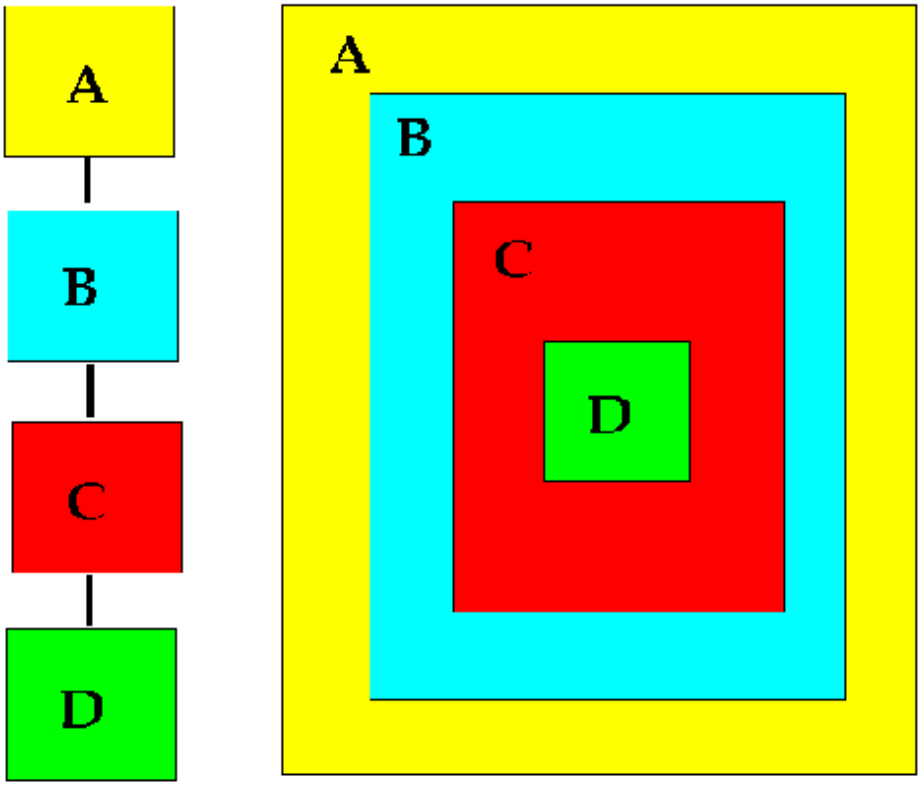
- Since transactional documents and data-intensive contexts tend to have more rules, their component models are more complex and there are more alternative document assembly models
- These alternate assembly models may differ in which information from the instance they present (they may be queries or views of the instance rather than a one-to-one rendering) and in the order or structure with which they present it.
- If the sequence is important it should be a component in the model and assembled in our logical documents (e.g. SequenceNumber in our Lecture Notes example)

The Rules of Assembly

- The rules represented in the component model must be followed during any document model assembly:
- Mandatory associations must be followed
- Mandatory components must be included
- Optional associations are followed if they meet the requirements for the context.
- Optional components are included if they meet the requirements for the context.
- Even if one role is the usual or canonical interpretation, it may not be a requirement for the context of this specific document assembly

---

Assembly Order and Containership



- The structural depth of the document model is determined by how many associations in the component model are followed
- The order in which associations are followed determines the nesting or container structure in the model

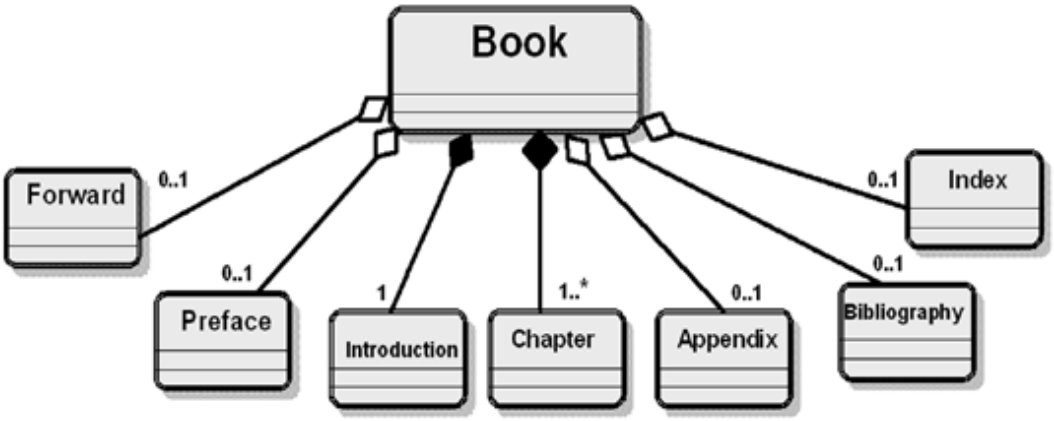
---

Document Model Assembly - Non-transactional Document Types

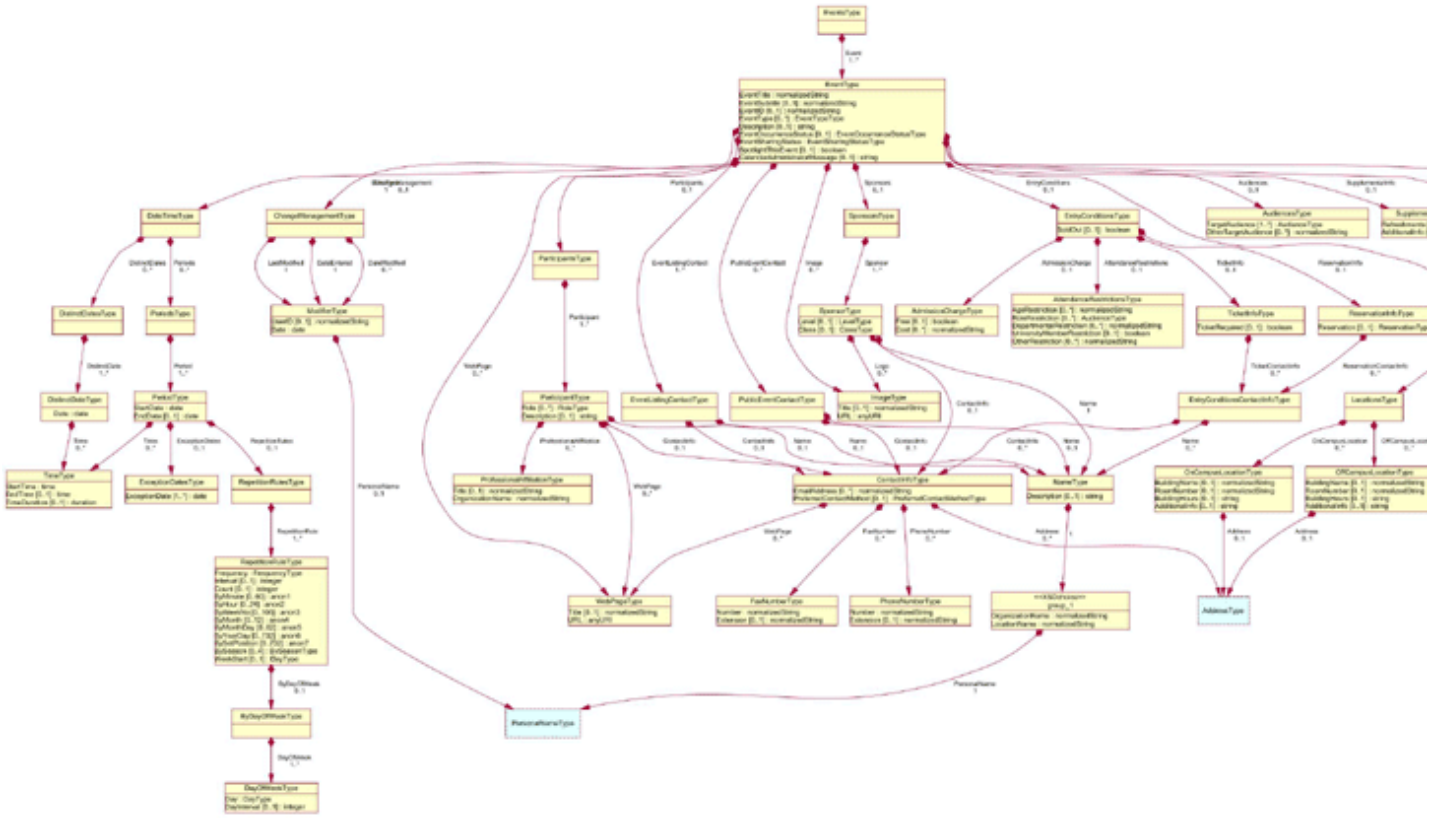
- Requirements for structural or presentation integrity may be more important than content constraints
- There are conventional assembly patterns for many types of documents (perhaps these can be viewed as default requirements)
- (Maler and el Andaloussi call this the "shape" of the document type)
- Some document types seem naturally "flat" – just 2-level deep "list of things" documents
- Sometimes documents can be arbitrarily deep with chapter, section, subsection, etc divisions but from a component type perspective this is a simple recursive structure with few or no content distinctions

---

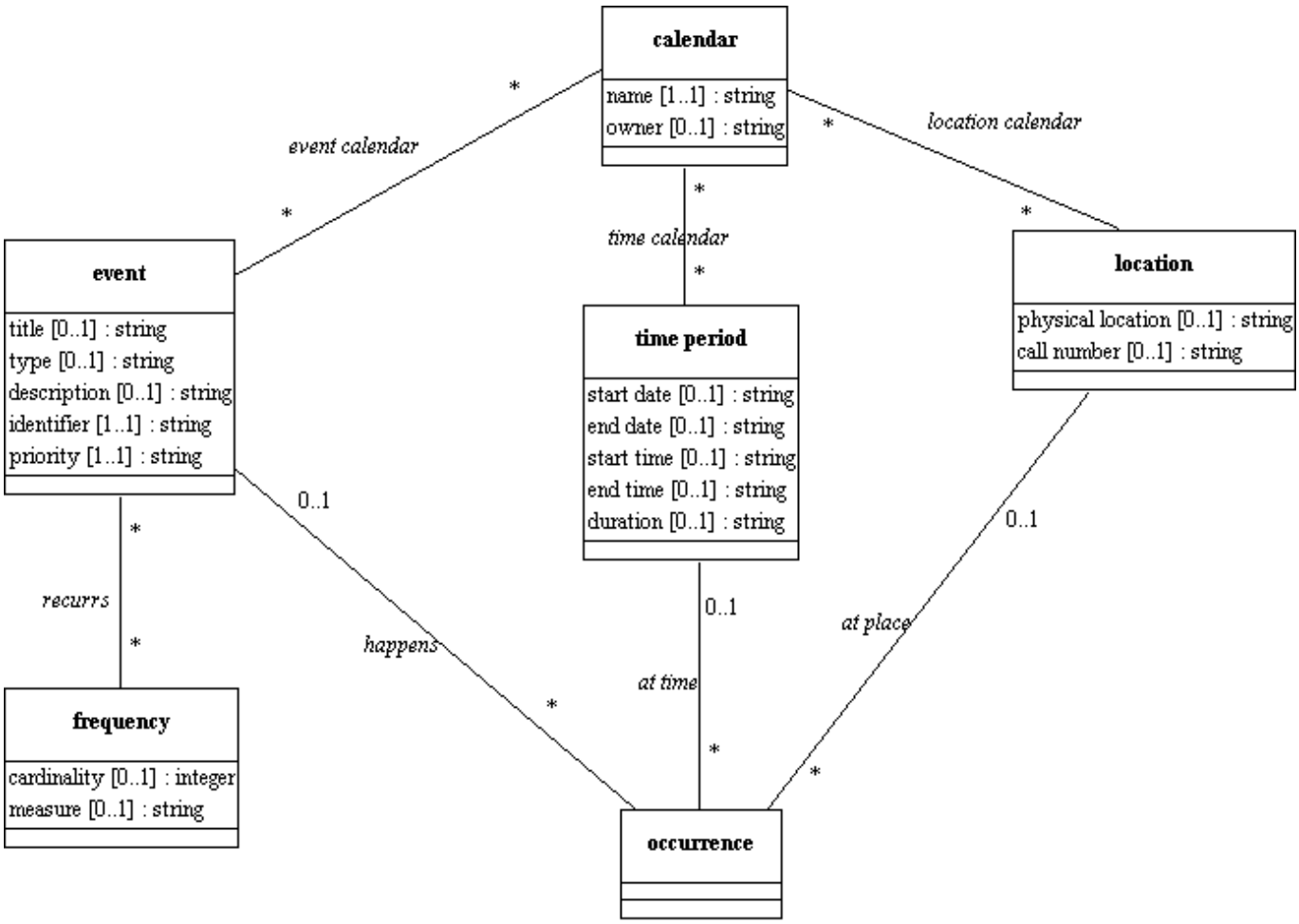
A Common Document Assembly Pattern



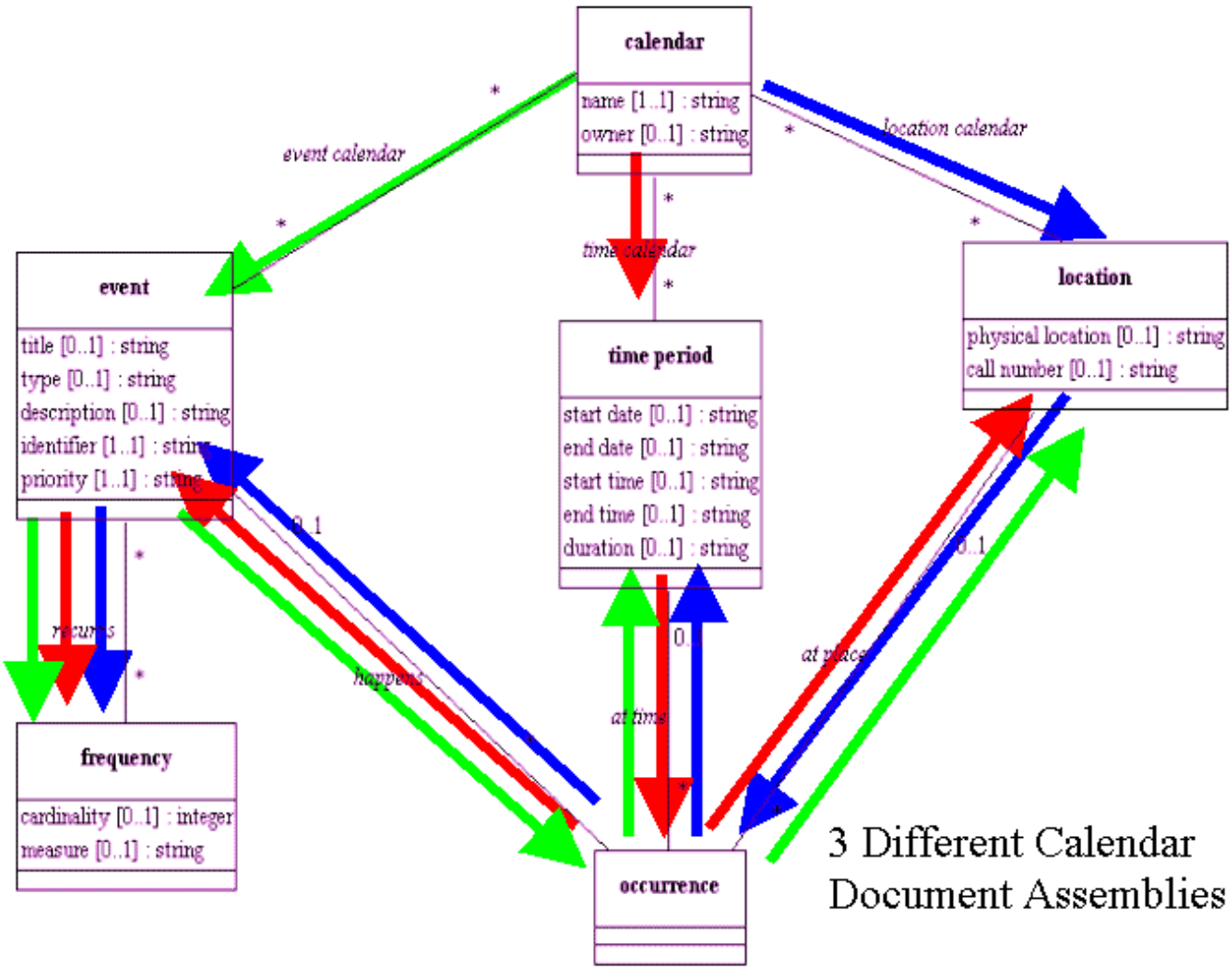
Conceptual Model of "Event" (Bloodworth & Glushko)



Simplified Conceptual Model



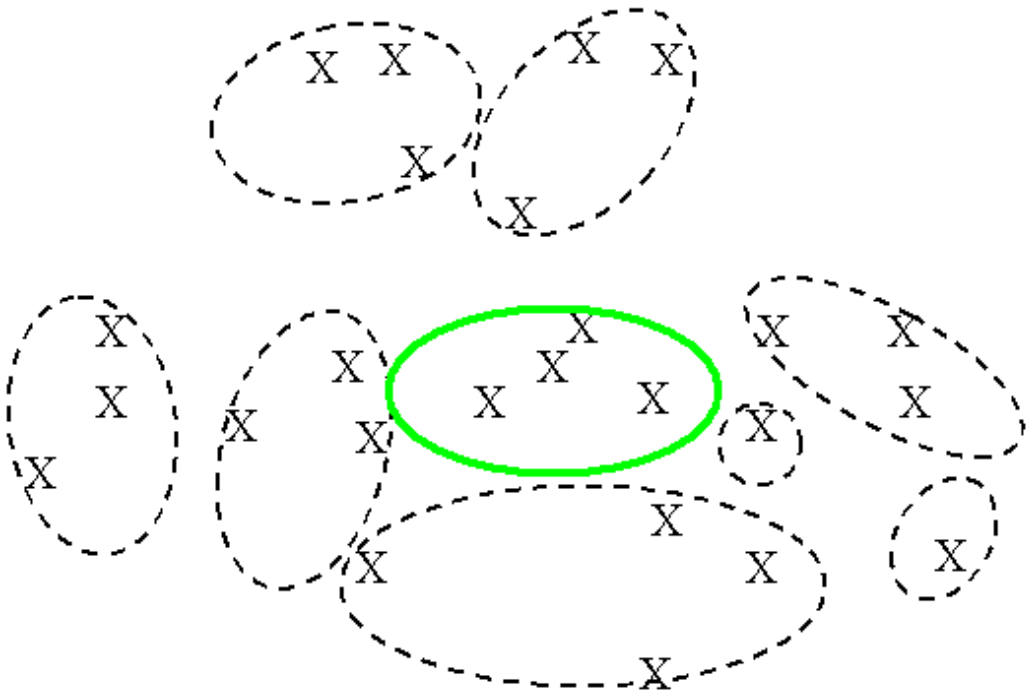
Assembling Different Calendar Models



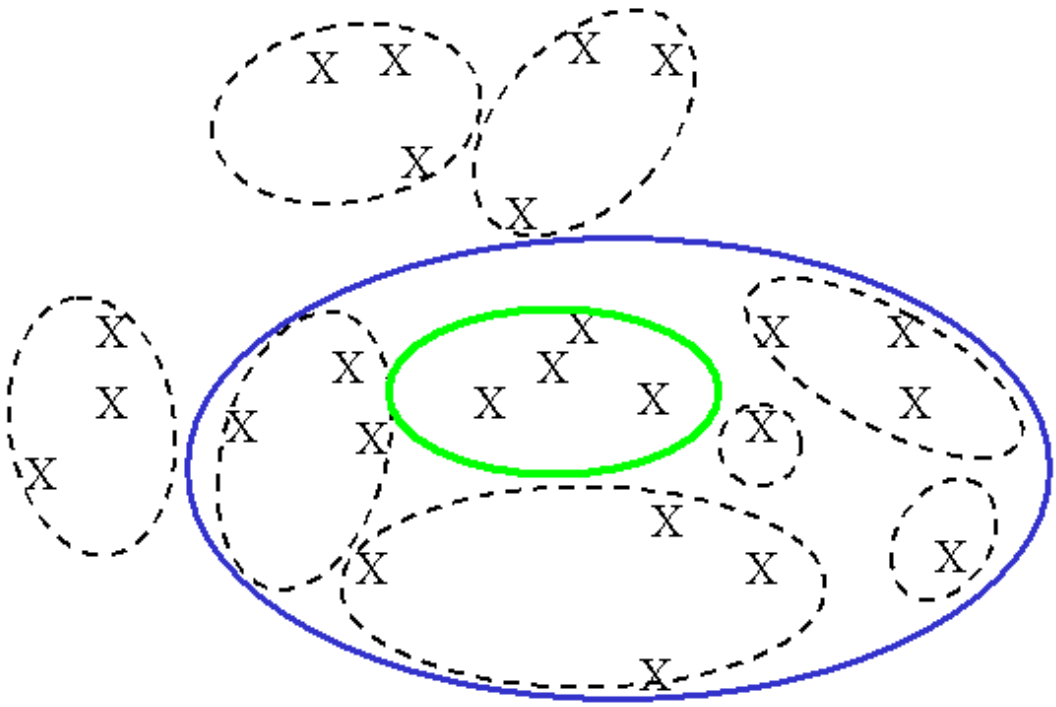
Motivating "Core and Contexts" Modeling and Assembly

- The "component model traversal" is a rigorous approach for assembling document models
- It is especially appropriate when you've been able to fully or mostly normalize the component model (because there are many constraints or rules about the components and their relationships)
- But the normalized model of a complex domain may be very granular with many small groups of components
- So we've developed a complementary conception of document model assembly that can improve the manageability and reuse of the model components – *core and contexts*
- The basic idea is that we're identifying components and assembling them from "the document down" rather than from "the components up"

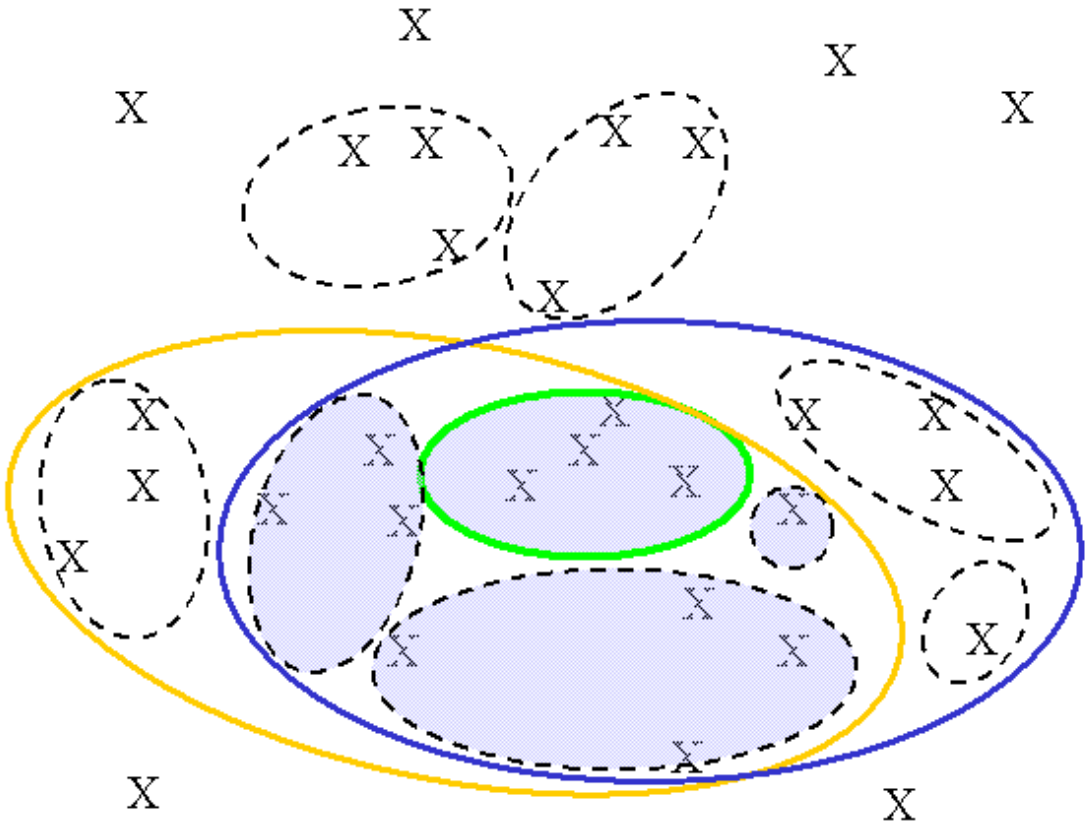
Core and Context Components



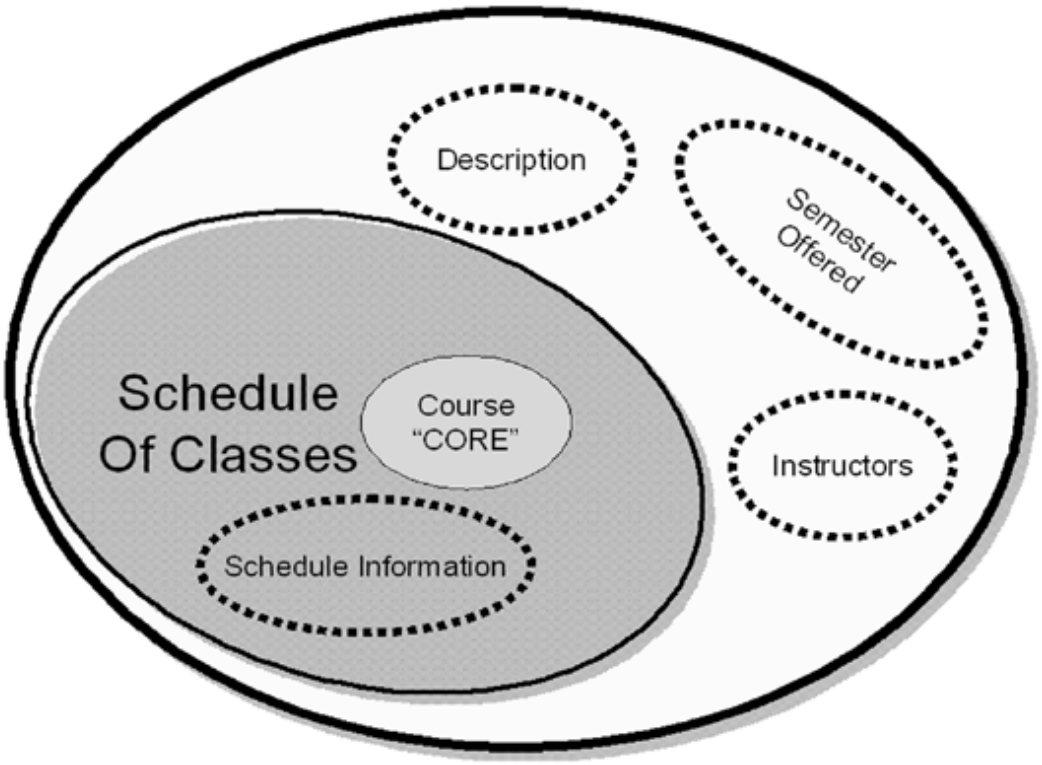
Assembly with Core and Contexts



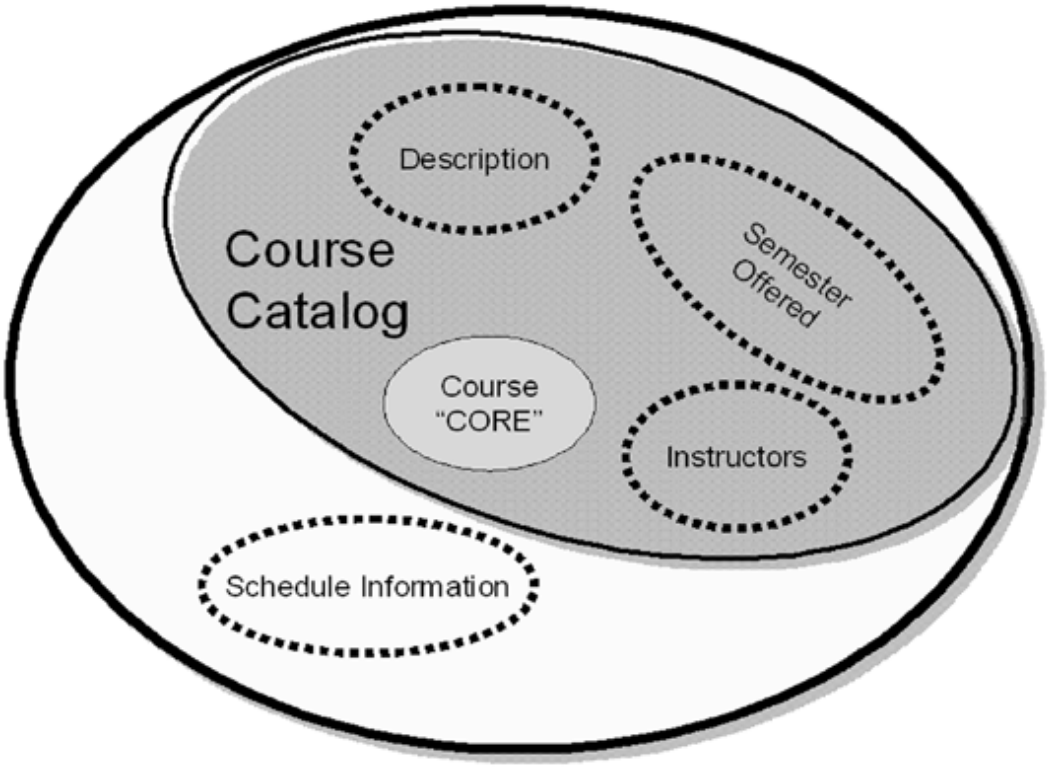
Reuse with Core and Contexts



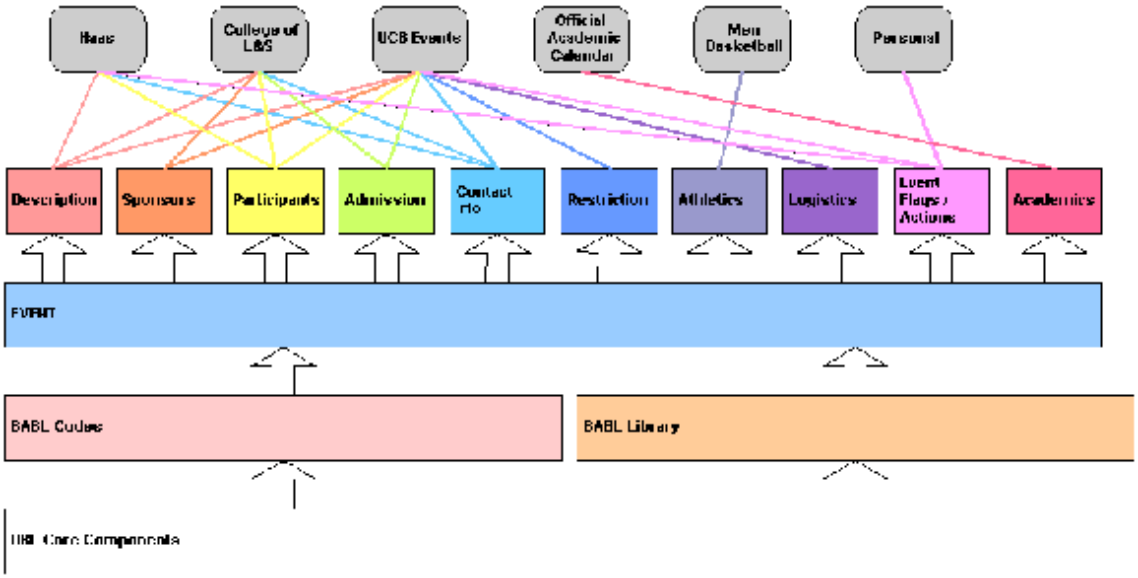
Modeling with Contexts in the Course "ecosystem" [1]



Modeling with Contexts in the Course "ecosystem" [2]



Modeling with Contexts for Event Calendars



Limitations of "Core and Contexts" Modeling

- It seems like a good idea to create the smallest possible core components and leave room for them to be customized or contextualized by additional components
- But the set of contexts that emerge is strongly shaped by the document types you are expecting to assemble, and some people object in principle to modeling shaped by implementation considerations
- Furthermore, the criteria or heuristics used to decide what "goes together" are informal and don't yield consistent results
- But it isn't a question of "either/or" here between the traversal and the c+c approach. Think of them as influences or philosophies or approaches for document assembly that you need to balance. Thinking of modeling and document assembly in different ways can result in a deeper understanding of why and how you got there

---

Summary

- "Document Engineering" is evolving as a new discipline for specifying, designing, and implementing the electronic documents that request or provide interfaces to business processes via Web-based services
- Best practices in Document Engineering require and reinforce the identification and reuse of patterns of information exchange from the perspective of business models, business processes, and document exchanges
- Doing business requires both "publication-like" document types like brochures and technical manuals and "transactional" documents like purchase orders and invoices – so we need analysis and design methods that work for both ends of this "Document Type Spectrum"

- Document Engineering can emphasize what these analysis and design approaches have in common rather than highlighting their differences
- The methodology we've systematized for Document Engineering seems to be interesting, learnable, and usable

---

Acknowledgments

- Much of this material comes from a book called *Document Engineering: Modeling for Business Informatics and Web Services* by Robert J. Glushko and Tim McGrath. MIT Press (2005)
- Three years of students at the University of California, Berkeley have contributed to its development through courses and research projects with the first author
- The methodology has been significantly refined through its use by the library content team of the Universal Business Language initiative, led by the second author

---