Avancier

Avancier Methods (AM)

CONCEPTS

Regular expressions in XSDs, Perl & JSP

Generalist architect training should

- Teach the universal truths of architecture description
  - Rather than specific documentation formats.
- Provide a general mental framework
  - into which architects can slot the variety of
    - system structures and behaviours
    - documentation notations
    - communication styles
    - technology types
  - that they meet

Architects should understand structures

- An architecture description is
- an abstraction in views or models
- of a concrete operational system.

- Those models define the structures and behaviours of the system.

Architects should understand structures in every domain

Avancier

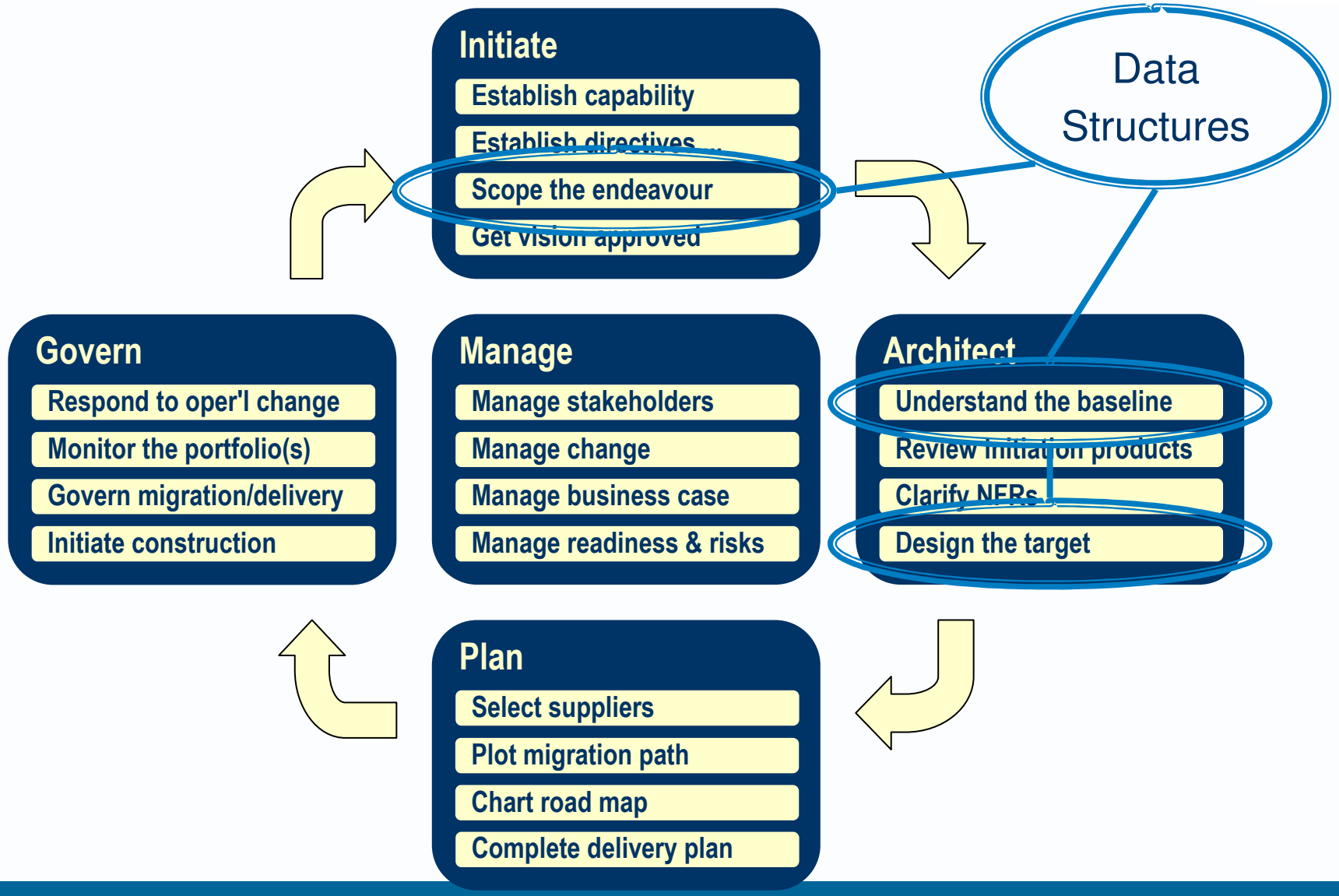
- Business structures
  - Organisation unit and business function structures
  - Process flow structures
- Data structures
  - Data store structures
  - Data flow structures
- Applications
  - Application inter-communication networks
  - The structure of application interactions within a process
- Infrastructure
  - Network structures

Architects should understand data structures

- An enterprise or solution architect should understand structural models of data processing systems
- The data architect is concerned with
  - data at rest - data stores
  - data in motion - data flows or messages
- The logical data structures of those data stores and data flows
- And other meta data
  - the individual data types that appear in data structures
    - (especially in canonical data models used in application integration).
  - other meta data such as CIA.

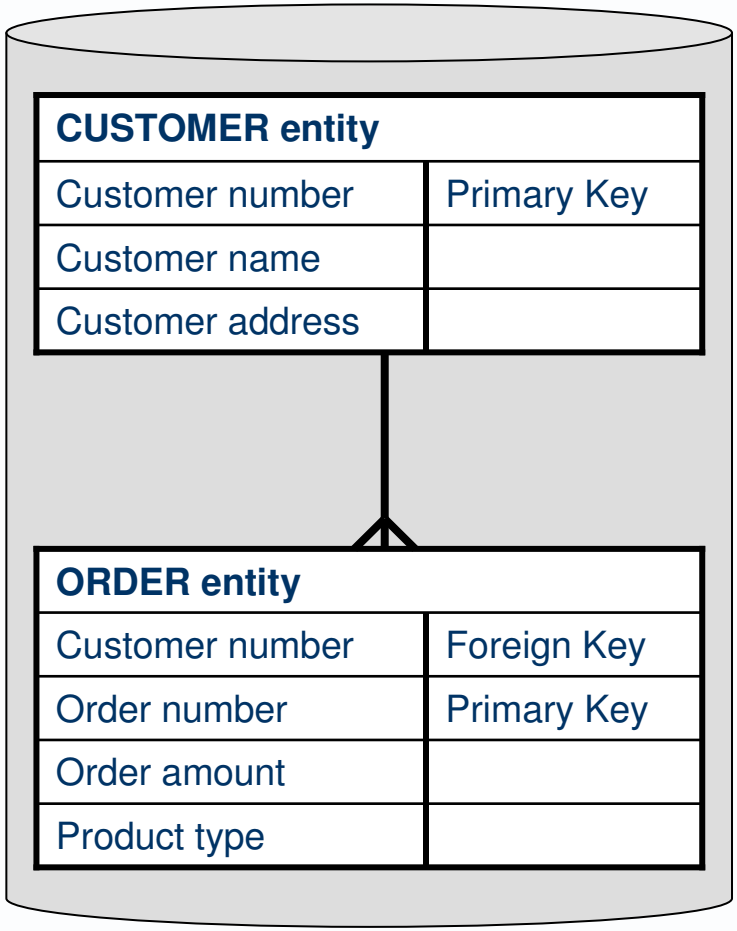
Where in the AM solution architecture process?

Avancier



Data store structures

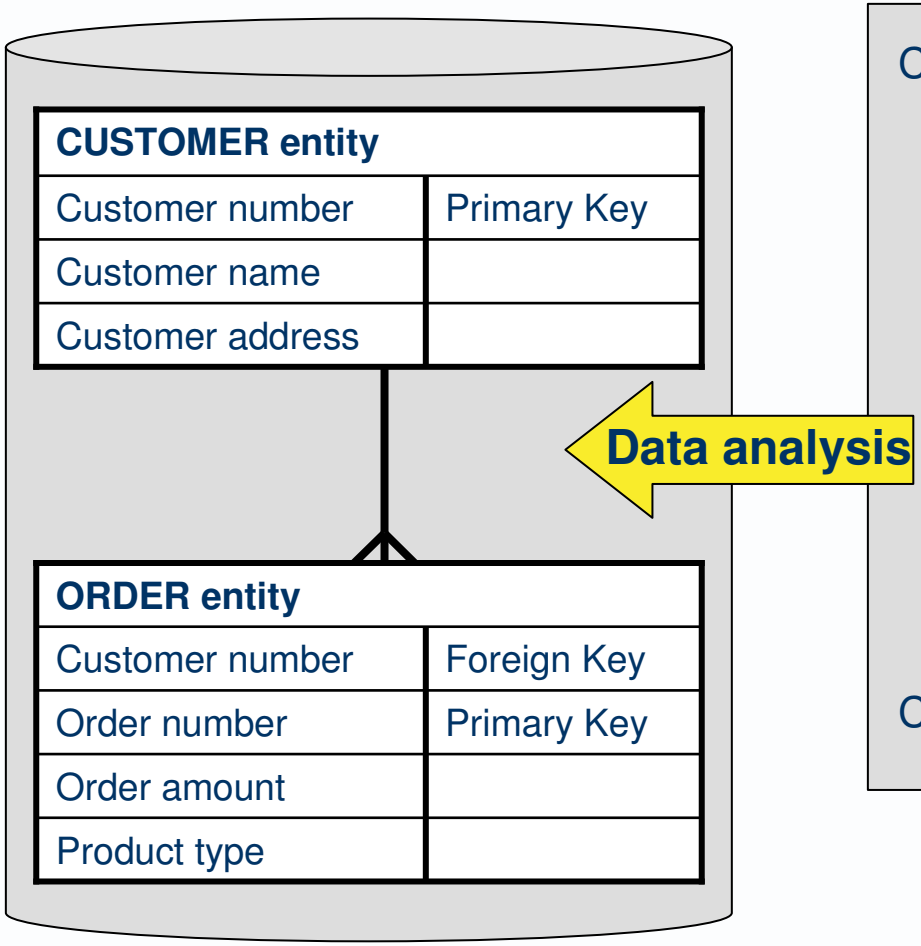
- The model of a data store's structure is a data model
  - that is, a network of entities, attributes and relationships.
- Data architects want to work at the level of the logical data model
  - rather than the physical database schema.



Q) How do we know what data to store?

Avancier

- A data store's content comes from Analysis of I/O data flows



Data analysis

Customer Order History SEQUENCE
- **Customer number**
- **Customer name**
- **Customer address**
- Set of Orders ITERATION
  - Order SEQUENCE
    - **Order number**
    - **Order amount**
    - **Product type**
  - Order END
- Set of Orders END

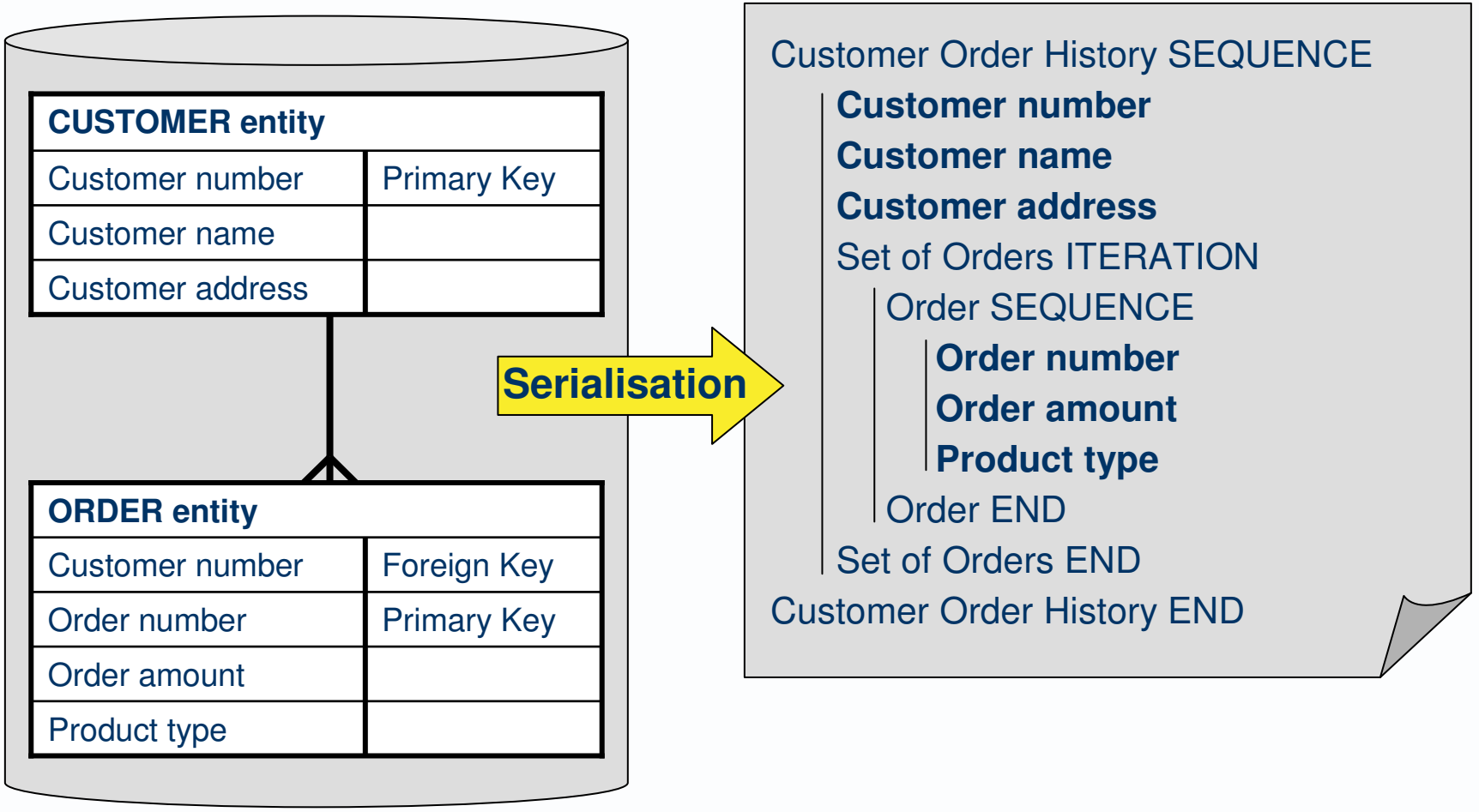
Customer Order History END

Q) How to generate a data flow from a data store?

Avancier

- Every data store can be serialised

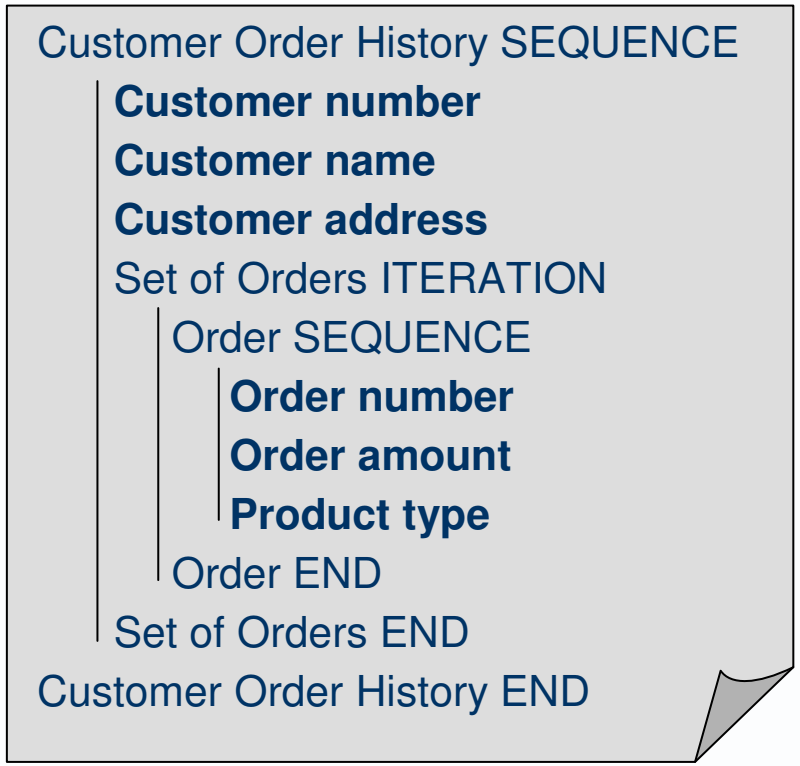
Into a hierarchical / sequential data flow



Data flows and logical structures

Avancier

- Data architects want to work at the level of the logical data flow structure
  - rather than physical data flow format

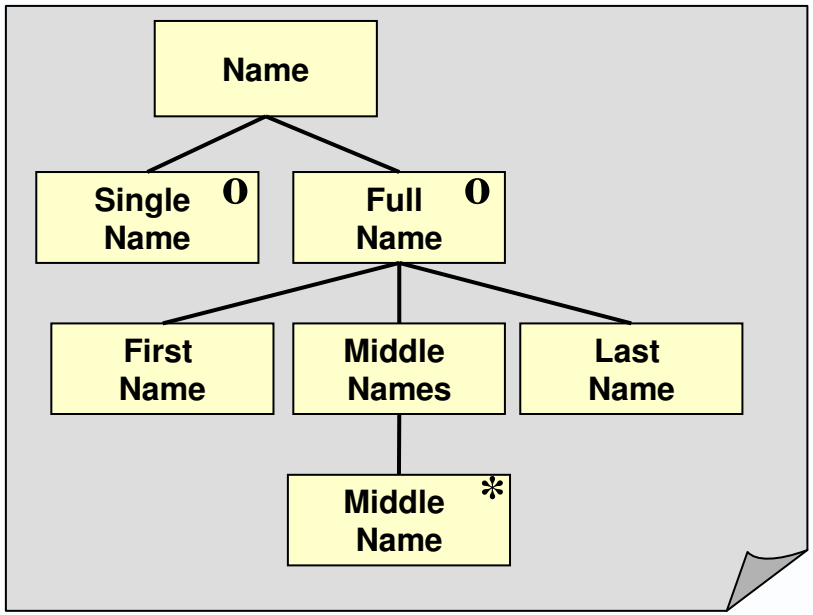


Data flow structures as regular expressions

Avancier

- The serialised structure of every data flow, be it
  - Serial file
  - Report
  - Message
  - Etc.

- Can be defined in the form of a regular expression
  - A hierarchical structure of elements arranged so that every element is:
    - part of a sequence
    - an option of a selection or
    - an occurrence of an iteration.

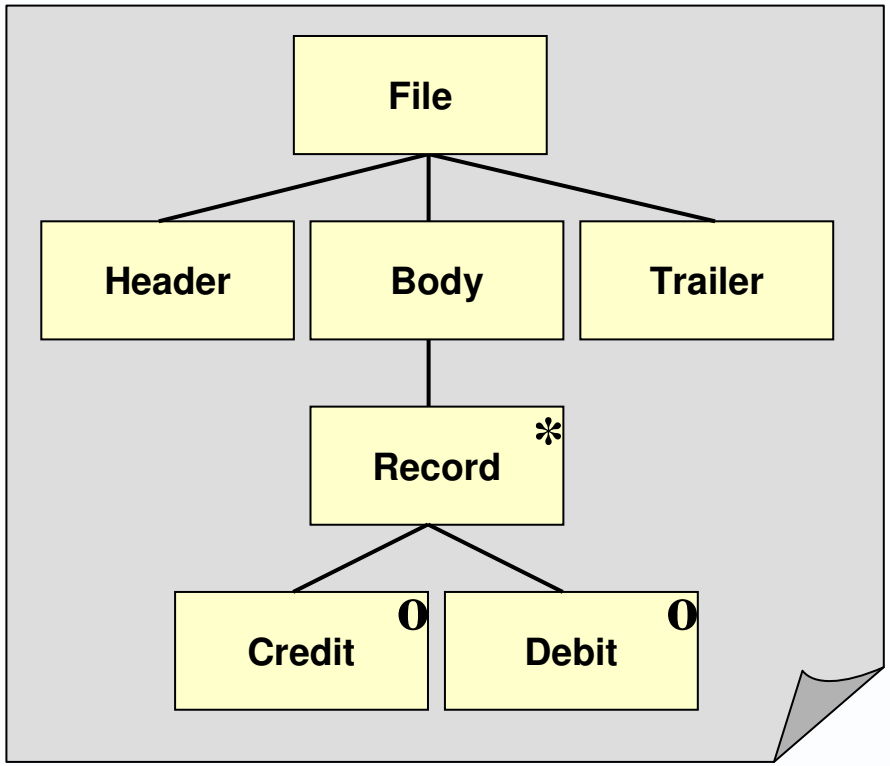


The data structure of a serial file

Avancier

File SEQUENCE
- **Header (b/f balance)**
- Body ITERATION
  - Record SELECT
    - **Credit**
    - Record OR
    - **Debit**
    - Record END
- Body END
- **Trailer (c/f balance)**

Full name END

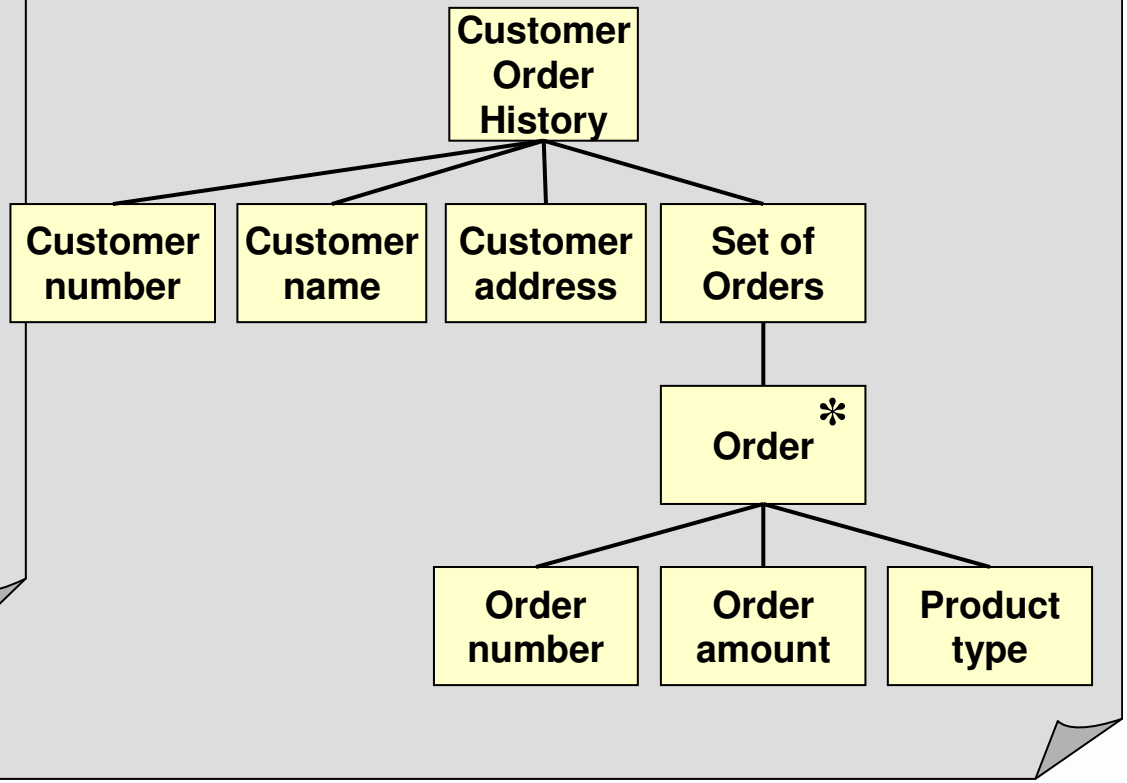


The data structure of a report

Avancier

Customer Order History SEQUENCE
- **Customer number**
- **Customer name**
- **Customer address**
- Set of Orders ITERATION
  - Order SEQUENCE
    - **Order number**
    - **Order amount**
    - **Product type**
  - Order END
- Set of Orders END

Customer Order History END



Data flow as XML document

Avancier

- XML evolved as a representation standard,
- but can also be considered in terms of classes and properties
- The purchase order example can be considered in terms of a
- “purchaseOrder” class and
- properties such as
  - “orderDate”,
  - “shipTo”,
  - “country”,
  - “name”,
  - “street”, etc.

```
<purchaseOrder orderDate="1999-10-20">
    <shipTo country="US">
        <name>Alice Smith</name>
        <street>123 Maple Street</street>
        <city>Mill Valley</city>
        <state>CA</state>
        <zip>90952</zip>
    </shipTo>
. . . .
</purchaseOrder>
```

XML Schema Primer
<http://www.w3.org/TR/xmlschema-0/>

Data flow structures in XSD

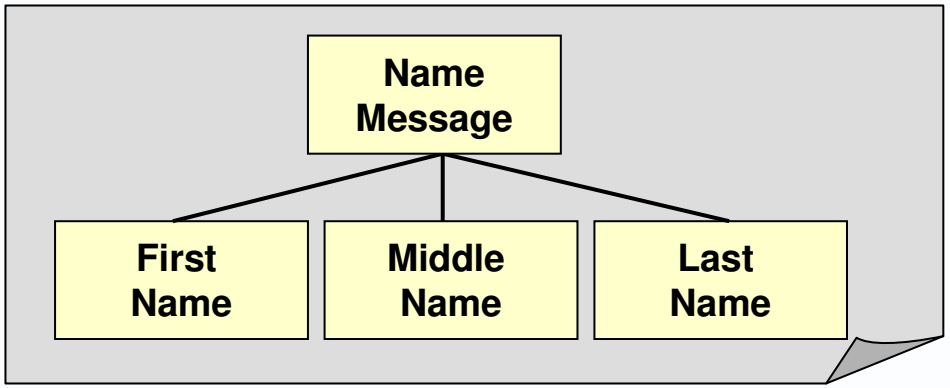
- People rarely document data flow structures in the ways shown on the last few slides
- Often, they draw examples: a report or screen format
- That is good
- But how to formally define the data flow that leads to it?
- And map the data items to data definitions?
- An XSD is one of many ways to represent the data content of data flows, including
  - data types
  - data flow structures (regular expressions)

The structure of a simple message in an XSD

Avancier

Many data flows are simple messages – simple sequences of data items

Name SEQUENCE
First Name
Middle Name
Last Name
Name END



xsd:sequence
xsd:element name="FirstName" type="Text"
xsd:element name="MiddleName" type="Text"
xsd:element name="LastName" type="Text"
/xsd:sequence

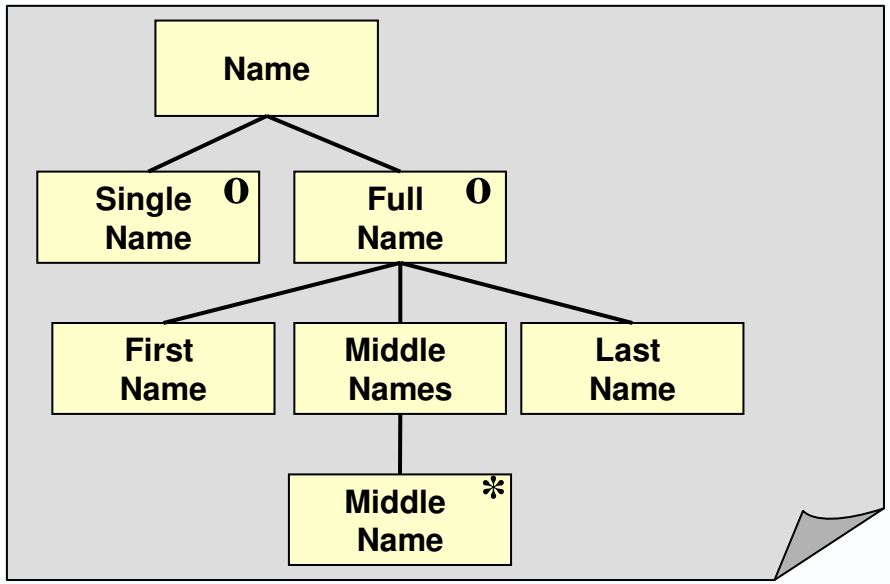
The structure of a complex data type in an XSD

Avancier

Name SELECT
- **Single Name**

Name OR
- Full name SEQUENCE
  - **First Name**
  - Middle Names ITERATION
    - **Middle Name**
  - Middle Names END
  - **Last Name**
- Full name END

Name END




```
xsd:choice
    xsd:element name="SingleName" type="Text" minOccurs="1" maxOccurs="1" /
    xsd:sequence
      xsd:element name="FirstName" type="Text" minOccurs="1" maxOccurs="1" /
      xsd:element name="MiddleName" type="Text" minOccurs="0" maxOccurs="unbounded"/
      xsd:element name="LastName" type="Text" minOccurs="1" maxOccurs="1" /
    /xsd:sequence
/xsd:choice
```

Footnote: regular expressions in program design

- The grammar of a regular expression is perhaps the most fundamental concept in all of computing.
- A computer cannot read or write a data flow unless the data flow is describable as a regular expression.
- The grammar is known to programmers from its use in
  - Pattern matching in character strings
  - Jackson structured program design.

The structure of a character string
As shown in a PERL-Compliant Regular Expression (PCRE)

Avancier

Space = \s

Character = Char (strictly = not\s, but I'm trying to make this readable)

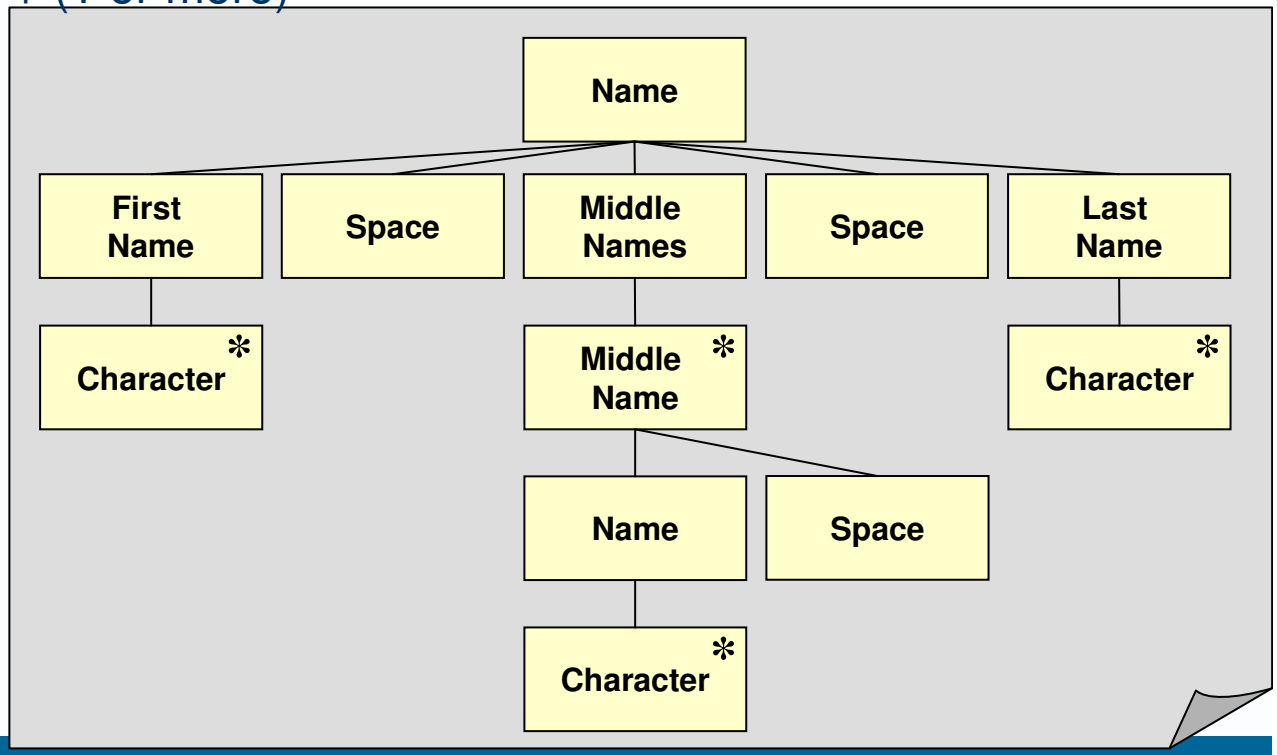
Element = [ ]

Selection of optional elements = [ABC]

Group = (?:)

Iteration = * (zero or more) + (1 or more)

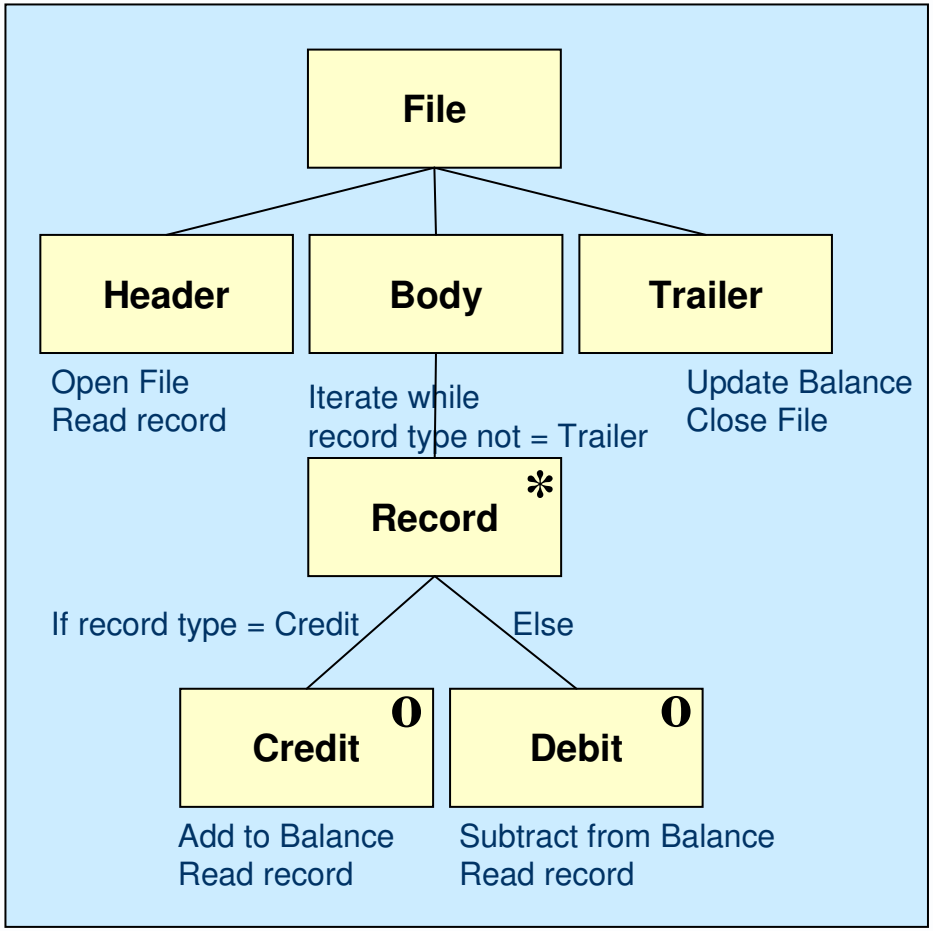
[Char]+ \s (?:[Char]+ \s)* \s [Char]+



The Jackson Structured Programming (JSP) technique

Avancier

1) define I/O data flow structures as regular expressions
2) form the process structure around the data flow structures
3) assign conditions to control the iterations and selections
4) assign operations to elements



Remember

Avancier

- The data architect is concerned with
  - data at rest - data stores
  - data in motion - data flows or messages
- The logical data structures of those data stores and data flows
- Other meta data
  - the individual data types that appear in data structures
    - (especially in canonical data models used in application integration).
  - other meta data such as CIA.
- Also
  - the distribution of data around the enterprise's estate
  - data mastering policy and design