The Structured-Element Object Model for XML

Oral Defense for the degree of Master of Philosophy

Presented by: Ma Chak Kei, Jacky

Committee Members
Prof. Y.S. Moon(Chairman)
Prof. Irwin King
Prof. Michael Lyu(Supervisor)

The Problem

- XML flexibly represents semi-structured data, however, it lacks the concepts of data encapsulation and object methods. Programmers have to identify each pieces of data to do the corresponding processing.
- There is a modeling gap between XML representation and OO programming representation.

Motivation

- XML data processing is important due to the increasing uses of XML in Internet applications and databases applications
- OO programming languages are widely used in developing for Internet applications
- Inadequate research on using XML data in OO programs
- XML Schema is popular in defining XML meta-data, but the current practice is limited to physical data validation
- Our objective is to construct a data model that better facilitate XML usage in OO programming, which supports data encapsulation and object methods. The model makes use of schema to define objects, and includes mechanisms for parsing and querying these objects

Contribution

- We propose the Structured-Element Object Model (SEOM) for handling XML data. The Model is extensible and flexible for using XML data in OO programming
- We propose the SEOM Schema for mapping XML Element data into SEOM class objects. The schema is generic to allow flexible representation of Element objects in XML data
- We propose a query wrapper technique for querying Structured-Element objects, which wraps the information of query details in an XML message
- We extend the XPath function to perform queries on Structured-Element objects
- We implement a Web-based SEOM Document Query System to demonstrate the feasibility of the model

Presentation Outline

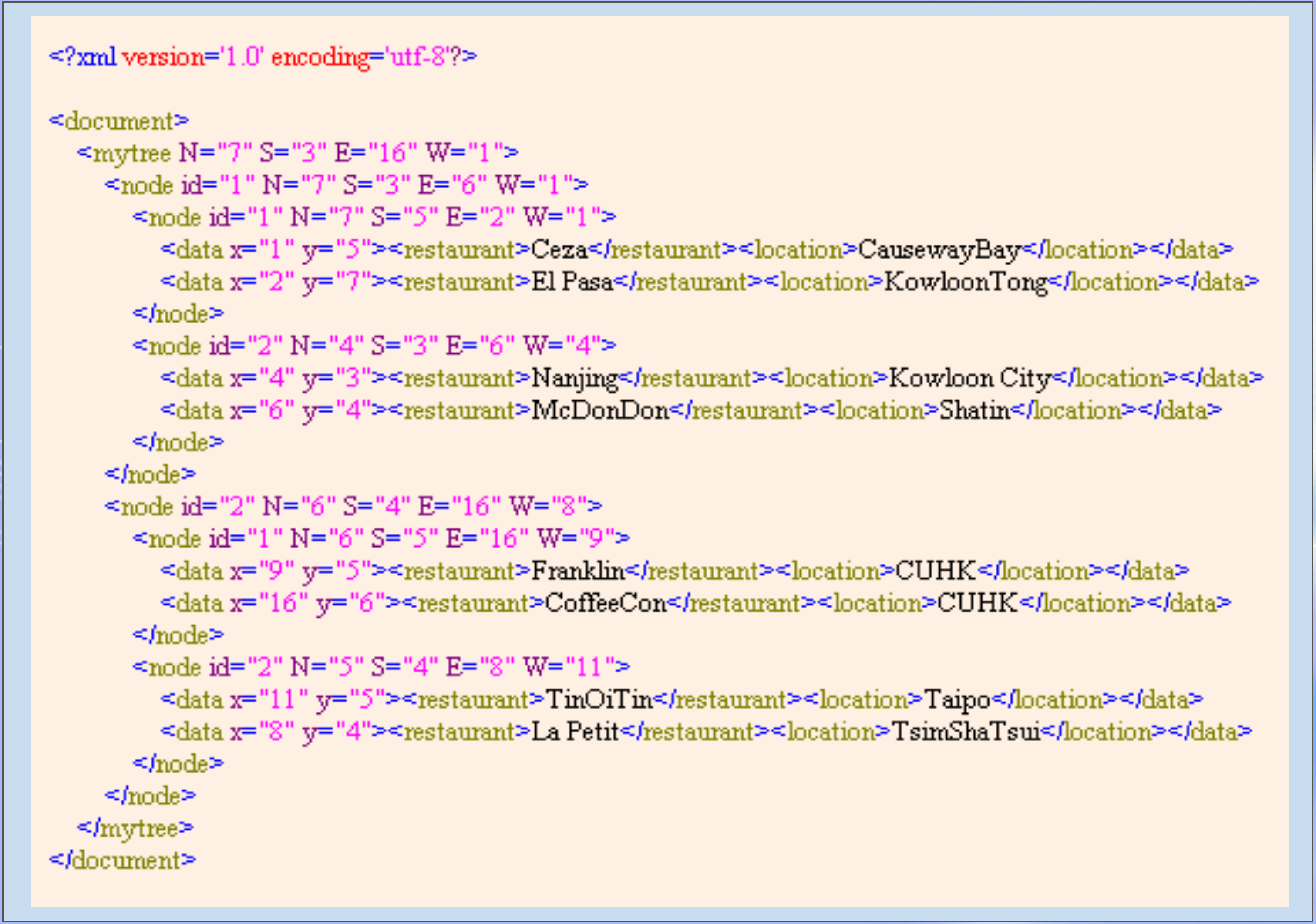
- Overview of XML and XML Data Modeling
  - Examples on XML Data Modeling
- The Structured-Element Object Model
  - Data Modeling
  - Schema Modeling
  - Classes Architecture
  - Parsing and Querying
- Web-Based SEOM Document Query System
- Evaluation & Conclusion

Overview of XML and XML Data Modeling

Overview of XML

- XML is a markup language for describing data. Its specification describes the XML data format and grammar. It can flexibly represent semi-structured data
- The family of XML technologies offers rich supporting functionalities, e.g.: XPath, Schema, DOM, namespaces, etc.
- XML provides a data exchange and representation standard
- XML favors cross-platform development of Internet applications

An XML Example



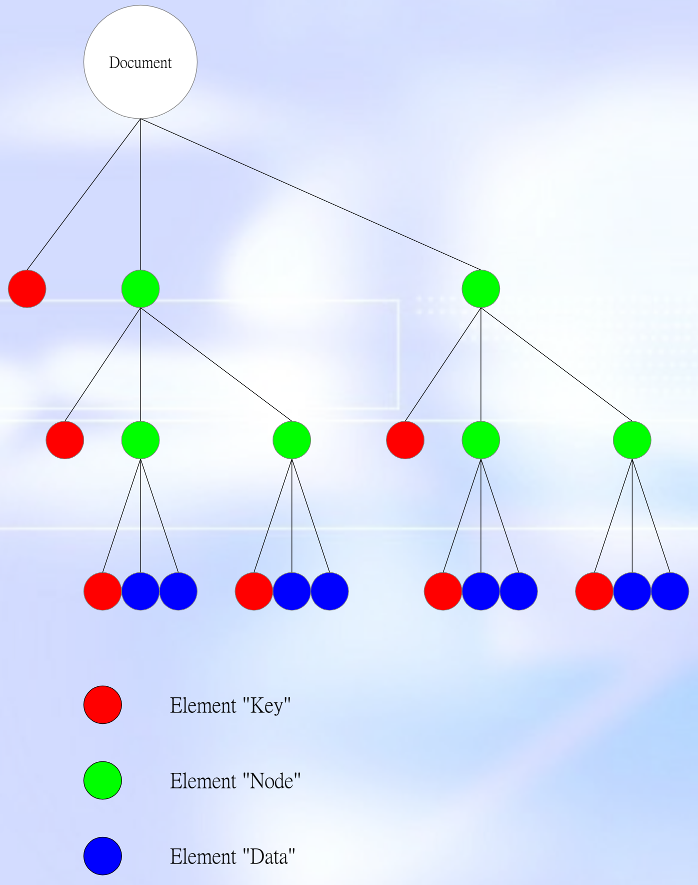
```xml
<?xml version='1.0' encoding='utf-8'?>

<document>
  <mytree N="7" S="3" E="16" W="1">
    <node id="1" N="7" S="3" E="6" W="1">
      <node id="1" N="7" S="5" E="2" W="1">
        <data x="1" y="5"><restaurant>Ceza</restaurant><location>CausewayBay</location></data>
        <data x="2" y="7"><restaurant>El Pasa</restaurant><location>KowloonTong</location></data>
      </node>
      <node id="2" N="4" S="3" E="6" W="4">
        <data x="4" y="3"><restaurant>Nanjing</restaurant><location>Kowloon City</location></data>
        <data x="6" y="4"><restaurant>McDonDon</restaurant><location>Shatin</location></data>
      </node>
    </node>
    <node id="2" N="6" S="4" E="16" W="8">
      <node id="1" N="6" S="5" E="16" W="9">
        <data x="9" y="5"><restaurant>Franklin</restaurant><location>CUHK</location></data>
        <data x="16" y="6"><restaurant>CoffeeCon</restaurant><location>CUHK</location></data>
      </node>
      <node id="2" N="5" S="4" E="8" W="11">
        <data x="11" y="5"><restaurant>TinOiTin</restaurant><location>Taipo</location></data>
        <data x="8" y="4"><restaurant>La Petit</restaurant><location>TsimShaTsui</location></data>
      </node>
    </node>
  </mytree>
</document>
```

Overview of XML Modeling

- A data model describes the structure, function, and constraints of the data; it affects how the data are manipulated in programs and how they are queried
- Two basic XML modeling:
  - Relational Model
  - Document Object Model
- Legacy application-specific data structures
  - E.g. various indexing trees
  - Similar modeling but proprietary implementation
  - Not interoperable, and difficult to maintain
  - Non-modular design and thus difficult to combine into more complex data structure

XML Modeling – Example

```
<key N="7" S="3" E="16" W="1"/>
<node id="1">
        <key N="7" S="3" E="6" W="1"/>
        <node id="1">
                <key N="7" S="5" E="2" W="1"/>
                <data x="1" y="5">itemA</data>
                <data x="2" y="7">itemB</data>
        </node>
        <node id="2">
                <key N="4" S="3" E="6" W="4"/>
                <data x="4" y="3">itemA</data>
                <data x="6" y="4">itemC</data>
        </node>
</node>
<node id="2">
        <key N="6" S="4" E="16" W="8"/>
        <node id="1">
                <key N="5" S="4" E="8" W="9"/>
                <data x="8" y="4">itemG</data>
                <data x="9" y="5">itemD</data>
        </node>
        <node id="2">
                <key N="6" S="5" E="11" W="16"/>
                <data x="11" y="5">itemF</data>
                <data x="16" y="6">itemE</data>
        </node>
</node>
```



Example – Relational Model

- Under the relational model, data are put into a table, regardless the difference in its role. Information are then retrieved using the "relation" of fields with SQL statements.
- Structural information in XML data is lost

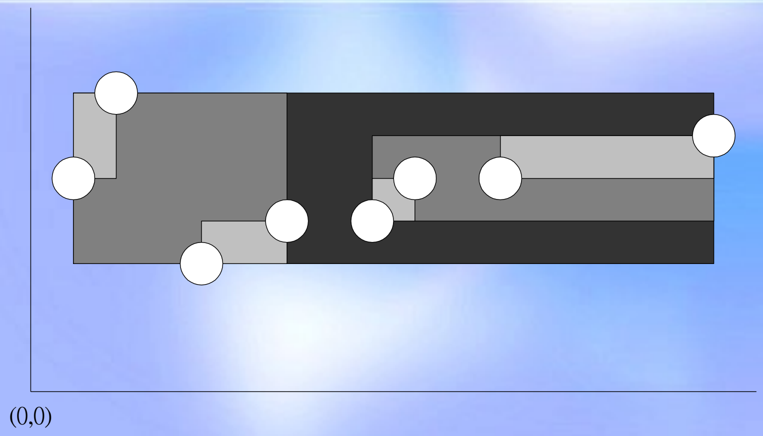
id1	N1	S1	E1	W1	id0	N0	S0	E0	W0	x	y	data
1	7	3	6	1	1	7	5	2	1	1	5	itemA
1	7	3	6	1	1	7	5	2	1	2	7	itemB
1	7	3	6	1	2	4	3	6	4	4	3	itemA
1	7	3	6	1	2	4	3	6	4	6	4	itemC
2	6	4	16	8	1	5	4	8	9	9	5	itemD
2	6	4	16	8	1	5	4	8	9	8	4	itemG
2	6	4	16	8	2	6	5	11	16	11	5	itemE
2	6	4	16	8	2	6	5	11	16	16	6	itemE

Example – Document Object Model

- DOM maintains the structure of XML data:
  - Retrieve the node “data” containing attribute “x=8”
    - //data[@x=‘8’]
  - Can retrieve parent-node, sibling-node, etc.
    - /node[1]/node[1]/key/following-sibling::*
- It is based on a generic tree structure, which does not require any assumption on the data
- Since it does not assume any knowledge on the data, all data are treated equally and little can be done on optimizing the manipulations

Example – Specific data structure, R-Tree

- For the same piece of XML data, if we know that represents an R-Tree structure, we can build a corresponding indexing structure in memory, and define meaningful methods on it:
  - Spatial Queries
    - Give me the point at (2,7)
    - Give me the point nearest to (4,4)
    - Give me the points bounded by (2,2) to (4,4)
  - Nearest Neighbor Search
    - Give me the point nearest to “itemB”



Comparison

- From the original XML data, we could not assume the semantics of the data:
  - We can do XML-based queries as in XPath
  - Or we can do queries based on the relationships in the tags as in the relational model
- From a model-based approach,
  - By using meta-data, we can define the model of a piece of XML data
  - We can define non-generic methods on data for a known model, such as the spatial queries in R-Tree model
  - Beside legacy data structure (like R-tree), there are also business data objects that may have its own data representation and manipulating/querying methods

The Structured-Element Object Model

SEOM – General Concepts

- Data Representation
  - Physical Data Representation – how the data are stored as files
    - Human-friendly: tables, hierarchical relationship
    - Machine-friendly: indexing trees
  - Logical Data Representation – how the data are represented as data objects
    - E.g. a "tree object", a business logic object, etc.
- Data Binding – the process of translating physical data representation to logical data representation
- Data Access – retrieve a particular record from a data object, e.g., search for a data point from a "search tree" object

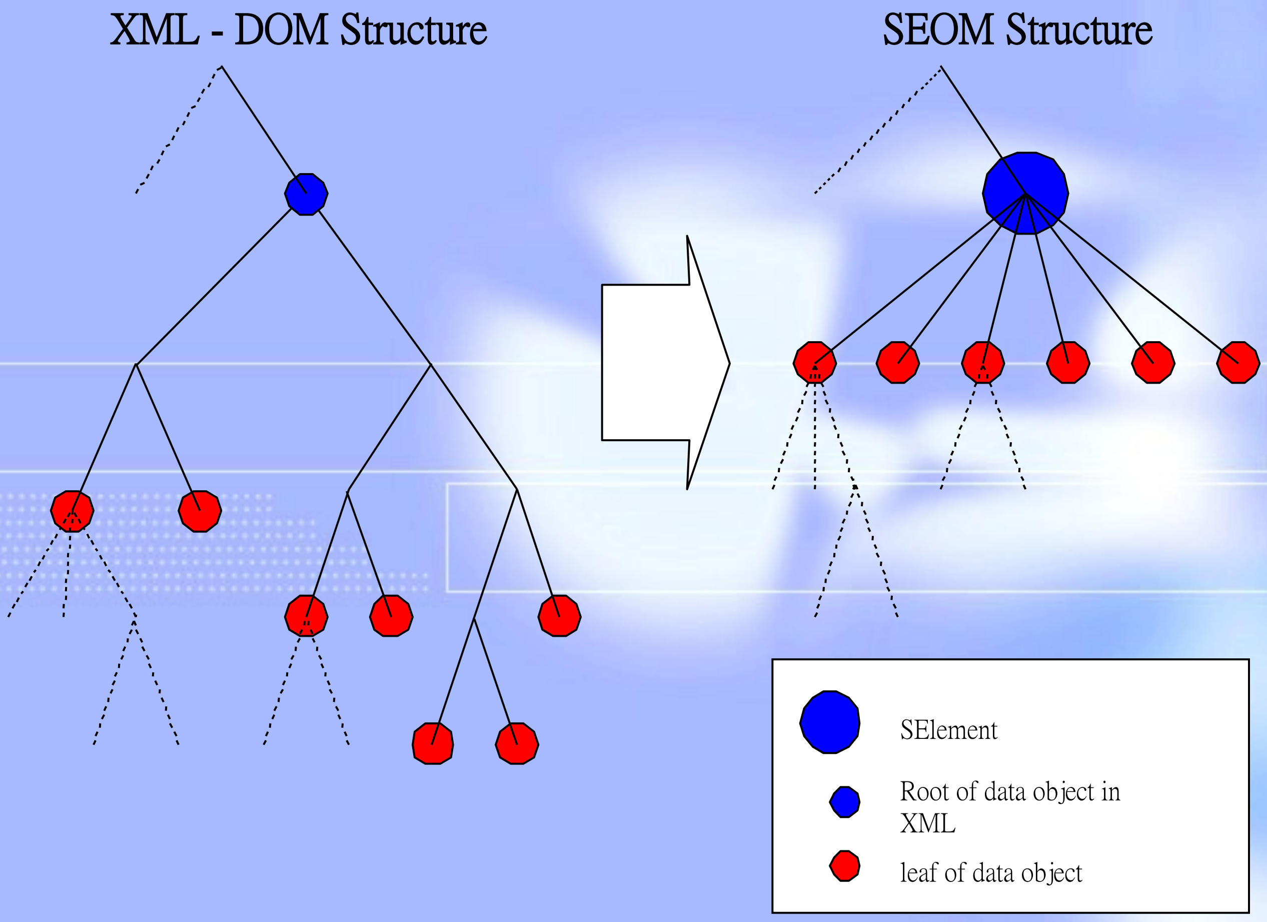
SEOM – Modeling

- Simple XML Data Model
  - Document, Element, Attribute, Character Data
- Document Object Model
  - Including Node, NodeList, AttributeSet for better management
- SEOM Data Model
  - Including an additional SElement type

SEOM – SElement

- Structured-Element (SElement) is an extension of DOM Element
- An data object encapsulates private information
- XML representation is defined by schema, including the internal branching and the child nodes
- Act as a mapping from data object root to leaf, with query method and query parameters as the selection criteria
- A query is modeled as a 3-dimension tuple:
  - [node, method, parameters]
  - node is specified by XPath
  - method is specified by a string value
  - parameters are specified in a multi-dimensional tuple, which varies for different methods

SEOM – SElement



SEOM – SElement

- Major methods (in addition to DOM Element's method):
  - `getTypeName()` – get the type name
  - `getSchema()` – get the schema document
  - `queryMethods()` – query for available query methods
  - `query()` – submit query to the SElement
  - `path()` – submit an XPath query to the SElement

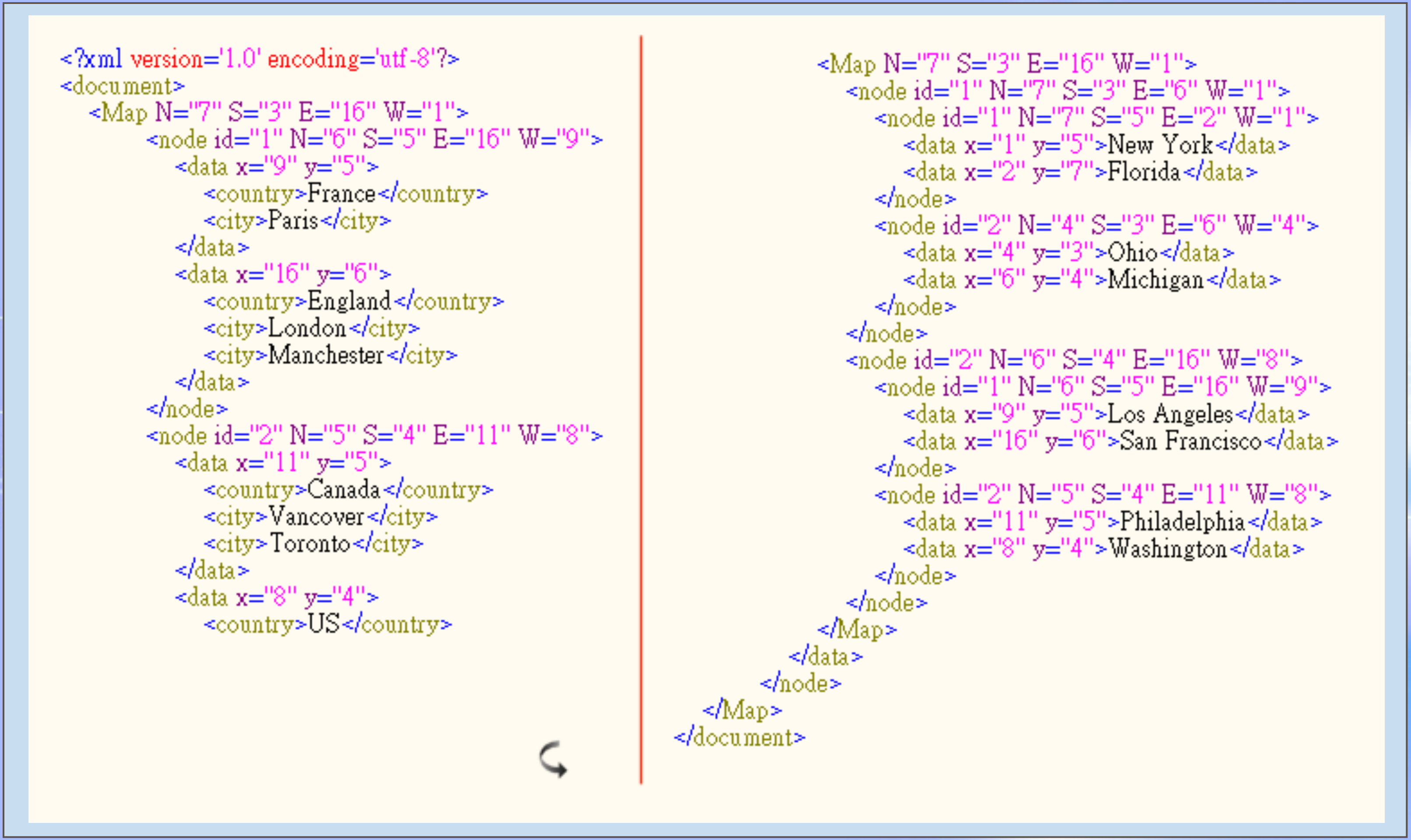
SEOM – Schema

- Provides meta data for describing the grammar of XML document to match a target model
- Defines:
  - the range of XML segments to be transformed from original DOM tree to SEOM SElement objects
  - the internal branching structure, e.g. number of branches, ordering, etc.
  - the data types of leaf nodes
  - the mapping from XML element values and attribute values to required parameters of the target model
- The extended schema is associated with a namespace with prefix "seom"

SEOM - Schema

- Major schema elements for SElement:
  - `seom:selement` – encapsulate an SElement definition
  - `seom:rootNode` – defines the root of the SElement
  - `seom:internalNode` – defines internal nodes of the SElement
  - `seom:leafNode` – defines leaf nodes of the SElement
  - `seom:attribute` – defines the attributes in root node, internal nodes and leaf nodes; may specifies model parameters by values or by referencing XML attribute values
  - `seom:value` – defines the structure under a root node, internal nodes and leaf nodes; may use XML schema elements to refine the constraints

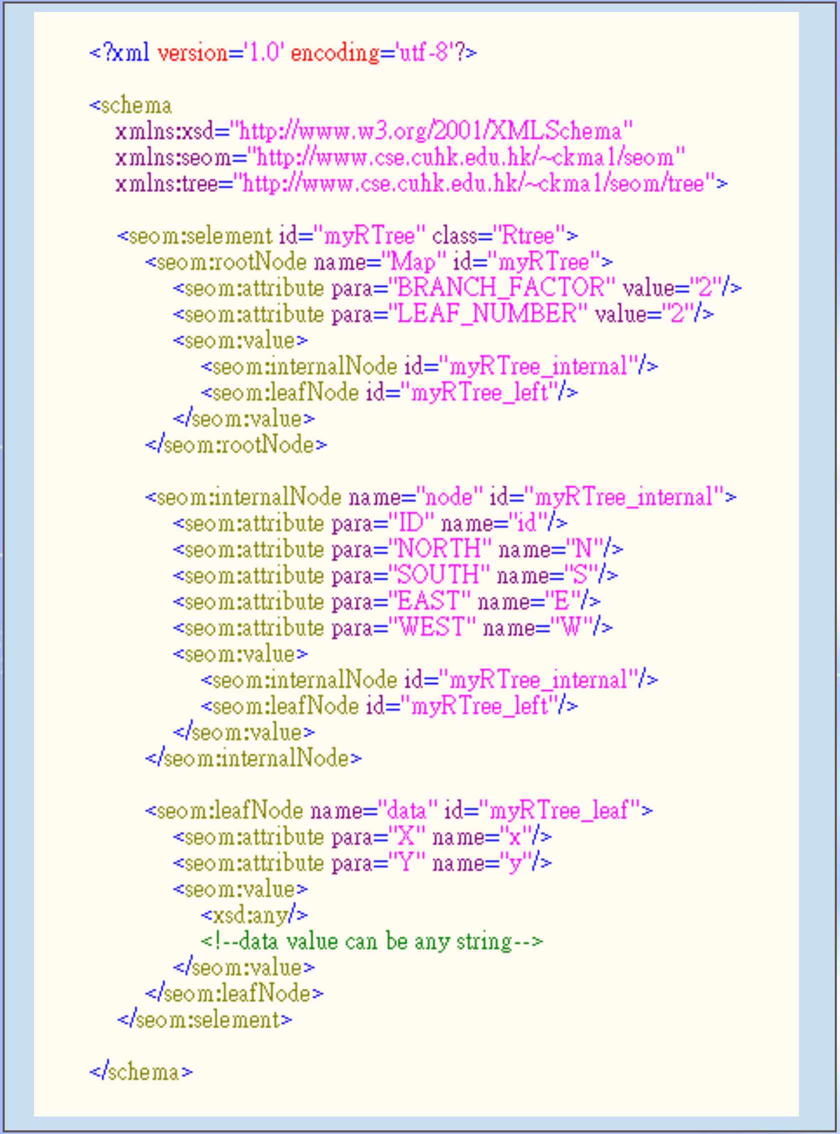
A Glance of XML Data



```xml
<?xml version='1.0' encoding='utf-8'?>
<document>
   <Map N="7" S="3" E="16" W="1">
         <node id="1" N="6" S="5" E="16" W="9">
            <data x="9" y="5">
               <country>France</country>
               <city>Paris</city>
            </data>
            <data x="16" y="6">
               <country>England</country>
               <city>London</city>
               <city>Manchester</city>
            </data>
         </node>
         <node id="2" N="5" S="4" E="11" W="8">
            <data x="11" y="5">
               <country>Canada</country>
               <city>Vancover</city>
               <city>Toronto</city>
            </data>
            <data x="8" y="4">
               <country>US</country>
```

```xml
                     <Map N="7" S="3" E="16" W="1">
                        <node id="1" N="7" S="3" E="6" W="1">
                           <node id="1" N="7" S="5" E="2" W="1">
                              <data x="1" y="5">New York</data>
                              <data x="2" y="7">Florida</data>
                           </node>
                           <node id="2" N="4" S="3" E="6" W="4">
                              <data x="4" y="3">Ohio</data>
                              <data x="6" y="4">Michigan</data>
                           </node>
                        </node>
                        <node id="2" N="6" S="4" E="16" W="8">
                           <node id="1" N="6" S="5" E="16" W="9">
                              <data x="9" y="5">Los Angeles</data>
                              <data x="16" y="6">San Francisco</data>
                           </node>
                           <node id="2" N="5" S="4" E="11" W="8">
                              <data x="11" y="5">Philadelphia</data>
                              <data x="8" y="4">Washington</data>
                           </node>
                        </node>
                     </Map>
                  </data>
               </node>
            </Map>
         </document>
```

A Glance of The Linked Schema



```
<?xml version='1.0' encoding='utf-8'?>

<schema
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:seom="http://www.cse.cuhk.edu.hk/~ckma1/seom"
  xmlns:tree="http://www.cse.cuhk.edu.hk/~ckma1/seom/tree">

  <seom:selement id="myRTree" class="Rtree">
    <seom:rootNode name="Map" id="myRTree">
      <seom:attribute para="BRANCH_FACTOR" value="2"/>
      <seom:attribute para="LEAF_NUMBER" value="2"/>
      <seom:value>
        <seom:internalNode id="myRTree_internal"/>
        <seom:leafNode id="myRTree_left"/>
      </seom:value>
    </seom:rootNode>

    <seom:internalNode name="node" id="myRTree_internal">
      <seom:attribute para="ID" name="id"/>
      <seom:attribute para="NORTH" name="N"/>
      <seom:attribute para="SOUTH" name="S"/>
      <seom:attribute para="EAST" name="E"/>
      <seom:attribute para="WEST" name="W"/>
      <seom:value>
        <seom:internalNode id="myRTree_internal"/>
        <seom:leafNode id="myRTree_left"/>
      </seom:value>
    </seom:internalNode>

    <seom:leafNode name="data" id="myRTree_leaf">
      <seom:attribute para="X" name="x"/>
      <seom:attribute para="Y" name="y"/>
      <seom:value>
        <xsd:any/>
        <!--data value can be any string-->
      </seom:value>
    </seom:leafNode>
  </seom:selement>

</schema>
```

SEOM – Implementation Issues

- A family of Java classes materialize the models. Instances of data objects are built from the classes with data from XML
- To construct an SEOM Document instance, it involves five types of classes:
  - Classes inherited from DOM
  - SEOM Document class
  - Abstract SElement class
  - Generic SElement class
  - Implement SElement classes
- Document processing
  - Parsing
  - Query

SEOM – Classes

- Classes inherited from DOM
  - Nodes, Elements, Attributes, etc.
  - Form the basic backbone of a DOM tree
- SEOM Document class
  - Corresponds to an XML document with additional interface for the SEOM-extended features
  - Constructor – take a DOM document and an XML Schema as parameters. Matching DOM elements will be send to a SElement constructor
  - Query – three query operations are implemented at this level:
    - DOM() – retrieve all direct children of a target node
    - Data() – retrieve the sub-tree of a target node in XML form
    - query() – generic interface for accepting user queries

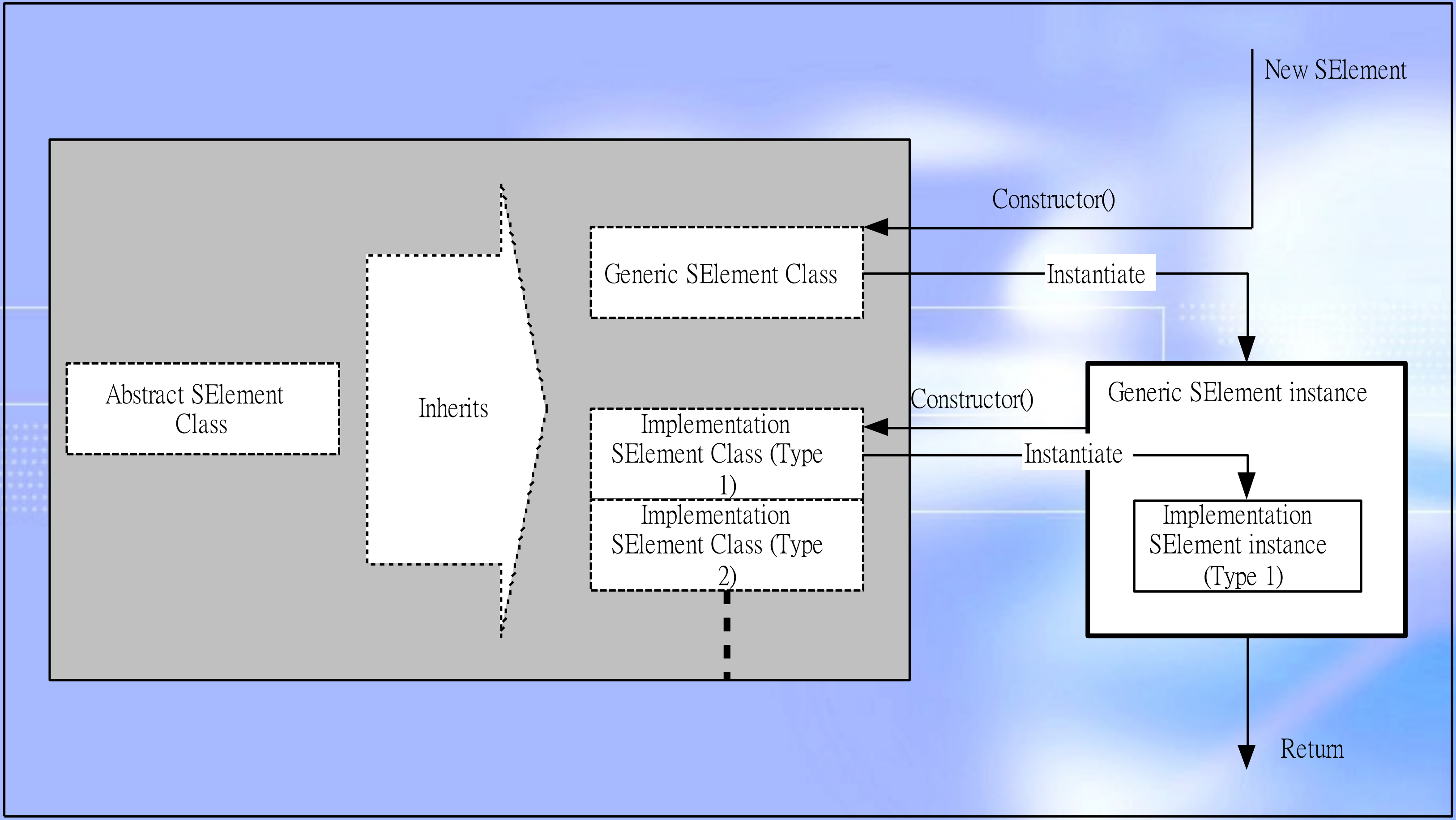
SEOM – Classes

- Abstract SElement Class
  - is the abstract superclass for all SElement data types
  - extends the DOM Element class to inherit its methods
  - defines abstract methods query() and queryMethod()
- Generic SElement Class
  - the only SElement class accessible to programmers
  - can instantiate an SElement object
  - wraps an implementation SElement class
    - fetch the needed implementation class
    - make the actual class transparent to the programmers
    - handle exceptions in creating SElement

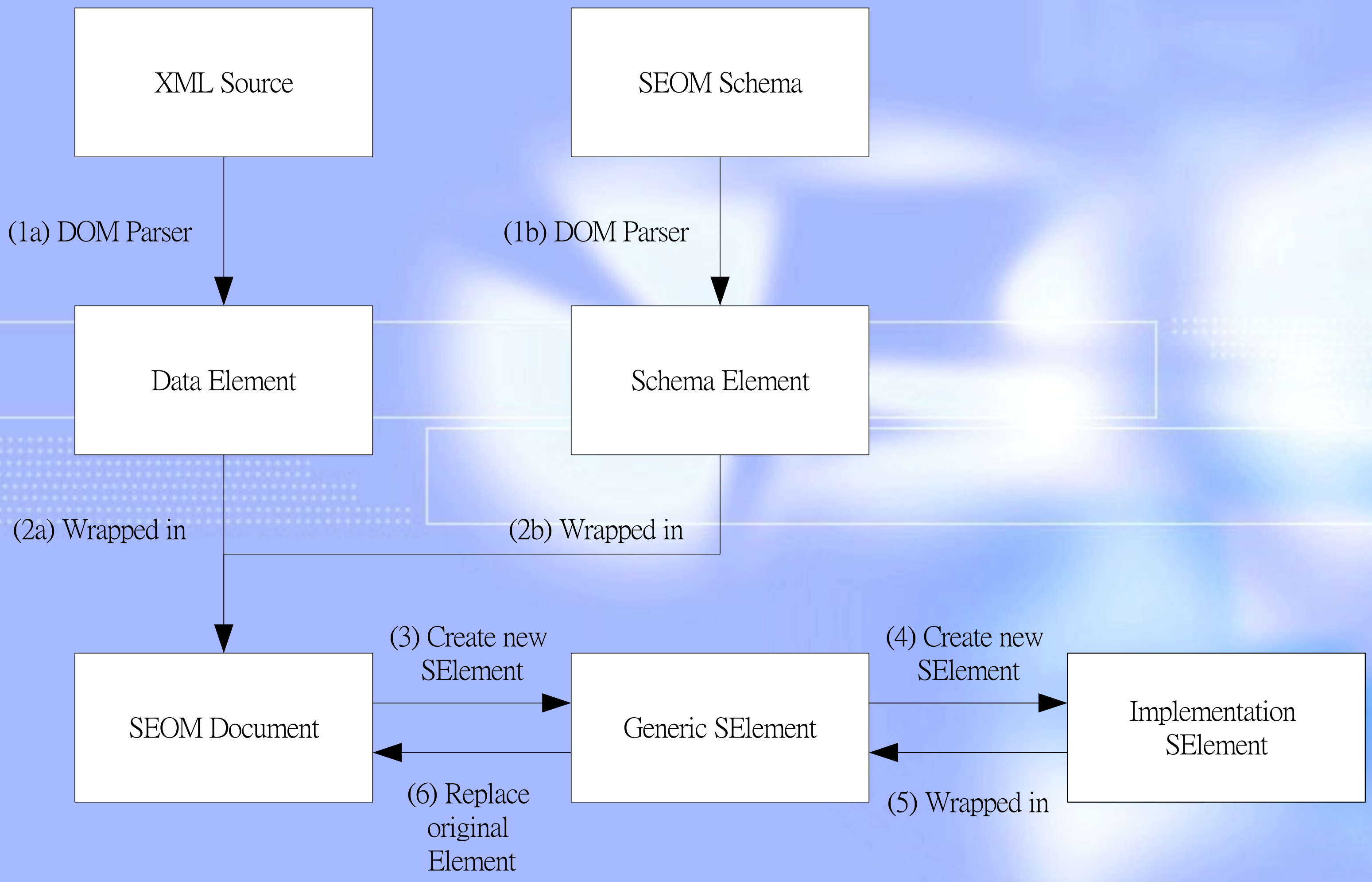
SEOM – Classes

- Implementation SElement Classes
  - It is indeed a group of classes, each class corresponds to one specific model
  - It has an internal data structure (instead of a DOM tree) to hold the data
  - It implements the constructor method to load data from XML to its internal data structure
  - It implements the query methods to fetch data from the internal data structure
  - Implemented classes:
    - An R-tree Class with exact search, range search, and k-nearest neighbor search
    - A Table Class with limited "select-from" clauses

SEOM – SElement Classes


New SElement
Constructor()
Generic SElement Class
Instantiate
Abstract SElement Class
Inherits
Constructor()
Generic SElement instance
Implementation SElement Class (Type 1)
Instantiate
Implementation SElement instance (Type 1)
Implementation SElement Class (Type 2)
Return

Parsing


XML Source
SEOM Schema
(1a) DOM Parser
(1b) DOM Parser
Data Element
Schema Element
(2a) Wrapped in
(2b) Wrapped in
(3) Create new SElement
(4) Create new SElement
SEOM Document
Generic SElement
Implementation SElement
(6) Replace original Element
(5) Wrapped in

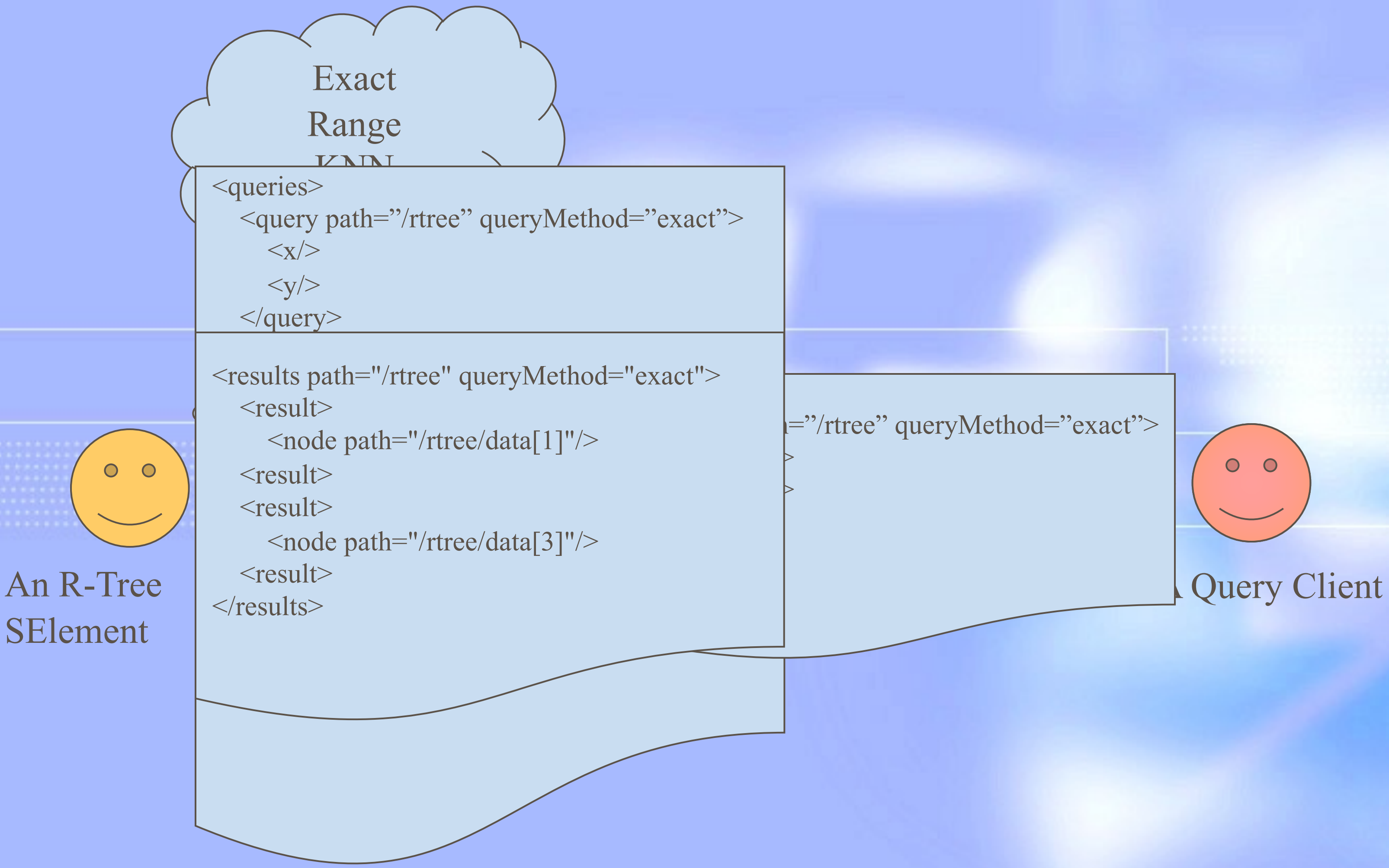
Query

- Two approaches: query wrapper and XPath
- Query Wrapper
  - Based on exchanging XML messages
  - Suitable for interactive querying between client and server
- XPath
  - Extended the W3C XPath with additional function
  - Suitable for pointing and referencing nodes for direct use

Query Wrapper

- Skeleton of query wrapper:
  - `<query path="" queryMethod=""></query>`
  - `path` specifies the target node using unique XPath expression
  - `queryMethod` specifies the name of method to be called
- Querying processes:
  - An query wrapper with empty `queryMethod` will retrieve the list of available query methods
  - The query wrappers for each query types will be returned as a `NodeList`
  - The user fill the parameters of a selected query wrapper and submit the query
  - Individual results are wrapped in <result> Elements; all <result> Elements are grouped under a single <results> Element; the results may in form of:
    - simple values (string, number)
    - composite values (XML Data)
    - child nodes of current SElement (wrapped in <node> element and specified in XPath)

Query Wrapper



```
<queries>
   <query path=”/rtree” queryMethod=”exact”>
      <x/>
      <y/>
   </query>
```

```
<results path="/rtree" queryMethod="exact">
   <result>
      <node path="/rtree/data[1]"/>
   <result>
   <result>
      <node path="/rtree/data[3]"/>
   <result>
</results>
```

```
=”/rtree” queryMethod=”exact”>
>
>
```

Extended XPath

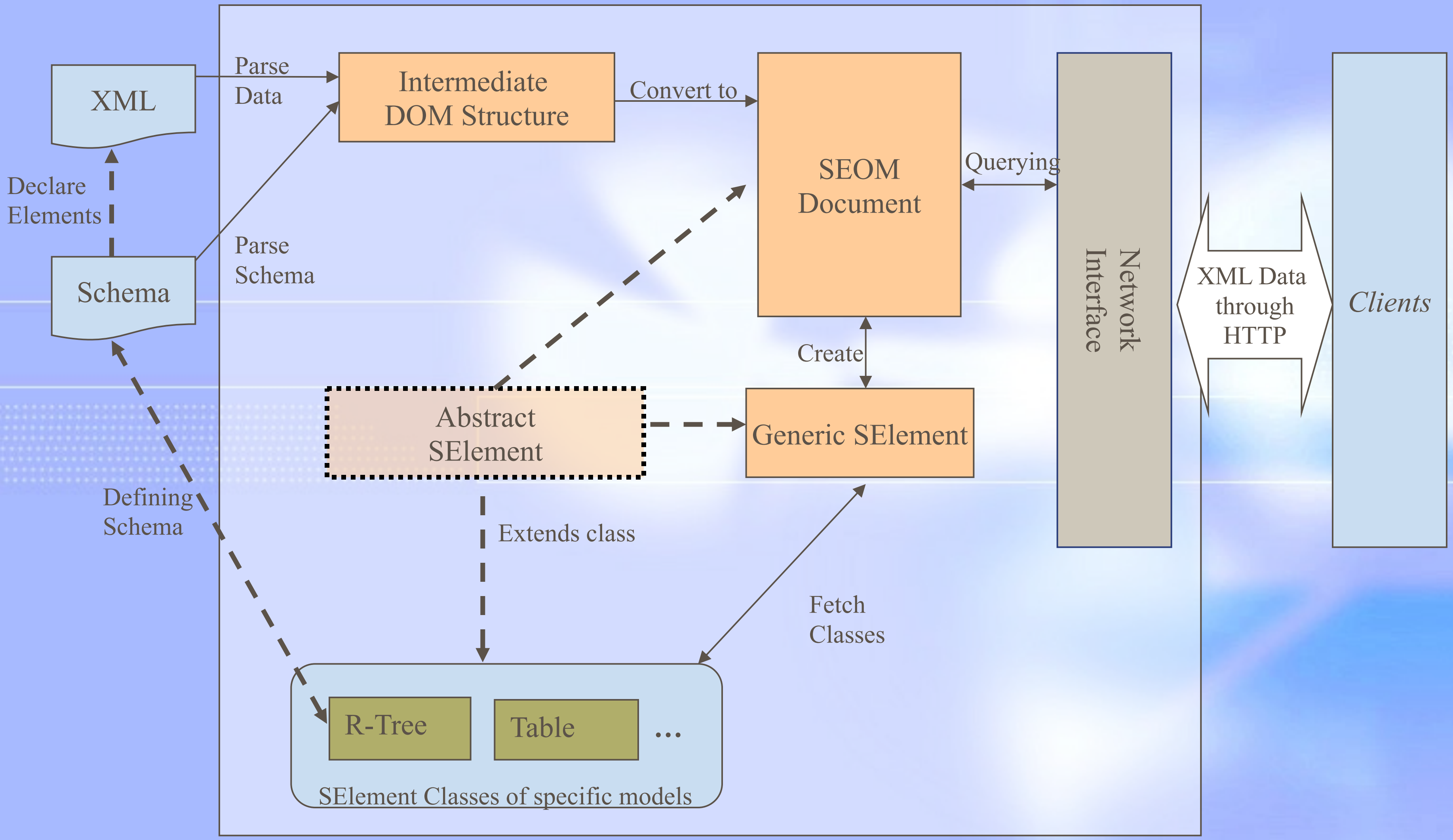
- In XPath, there are functions for manipulating strings, numbers, and Booleans. We introduce a function to allow queries to be made to SElements.
- The basic query form is to specify a target SElement node, a method name, and a set of parameters in name-value pairs, e.g.
  - query("/document/selement", "exact", "x=3", "y=4")
- A more common use of XPath function is to select nodes with predicate
  - The predicate is added as a filter to the context node, i.e., the leaf nodes of SElement
  - /document/selement/data[query("exact", "x=3", "y=4")]
  - The function itself results in a boolean value. It takes the context position implicitly and evaluates the query according to that

A Web-Based SEOM Document Query System

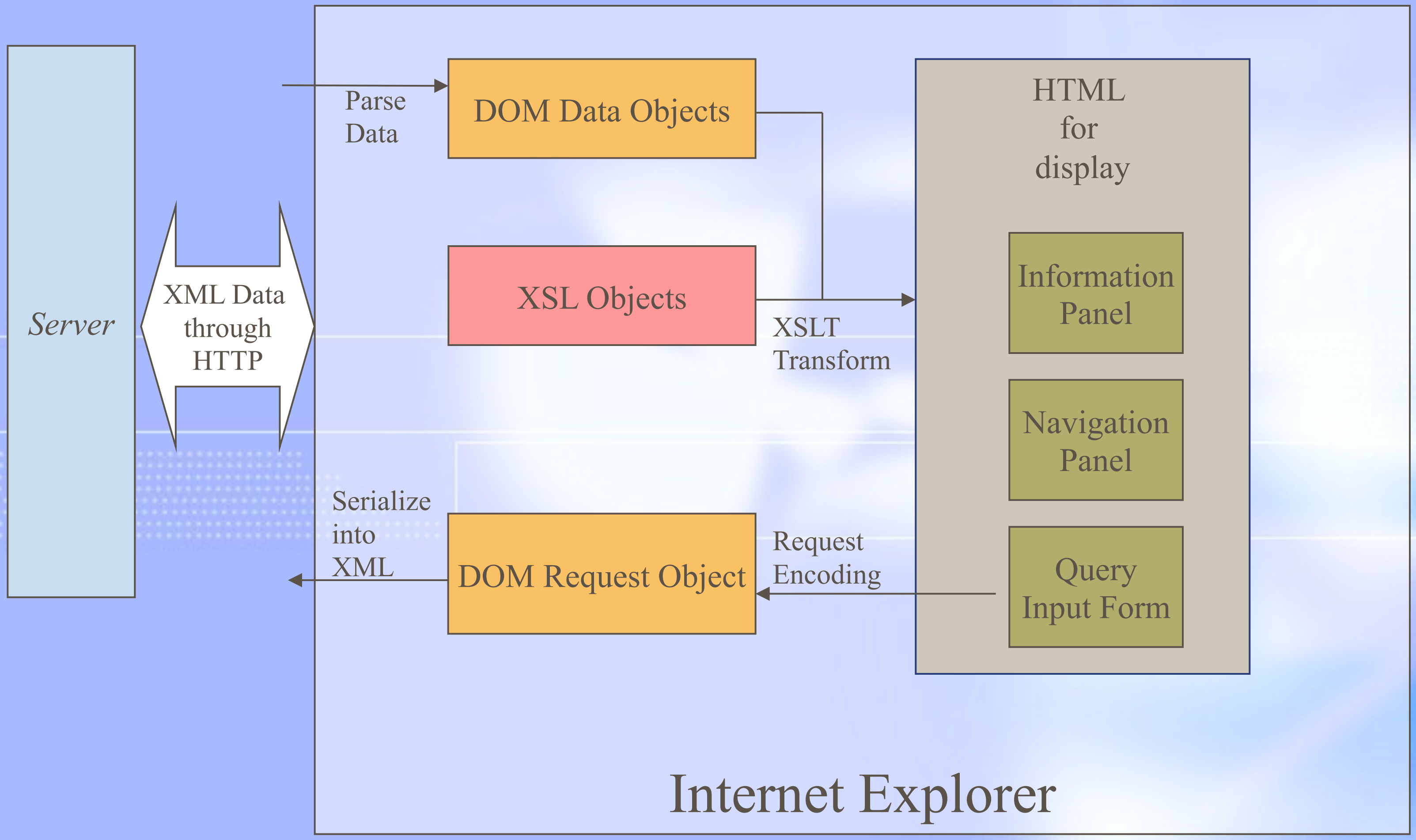
Web-Based SEOM Document Query System

- Objective
  - To demonstrate the feasibility of our model, including the schema, the parsing process, as well as the query process
  - To illustrate how it assists in querying XML data
  - To facilitate as the platform for testing the implementation of arbitrary structured models
- Implemented with JDK1.4
- Available models: R-Tree, Table

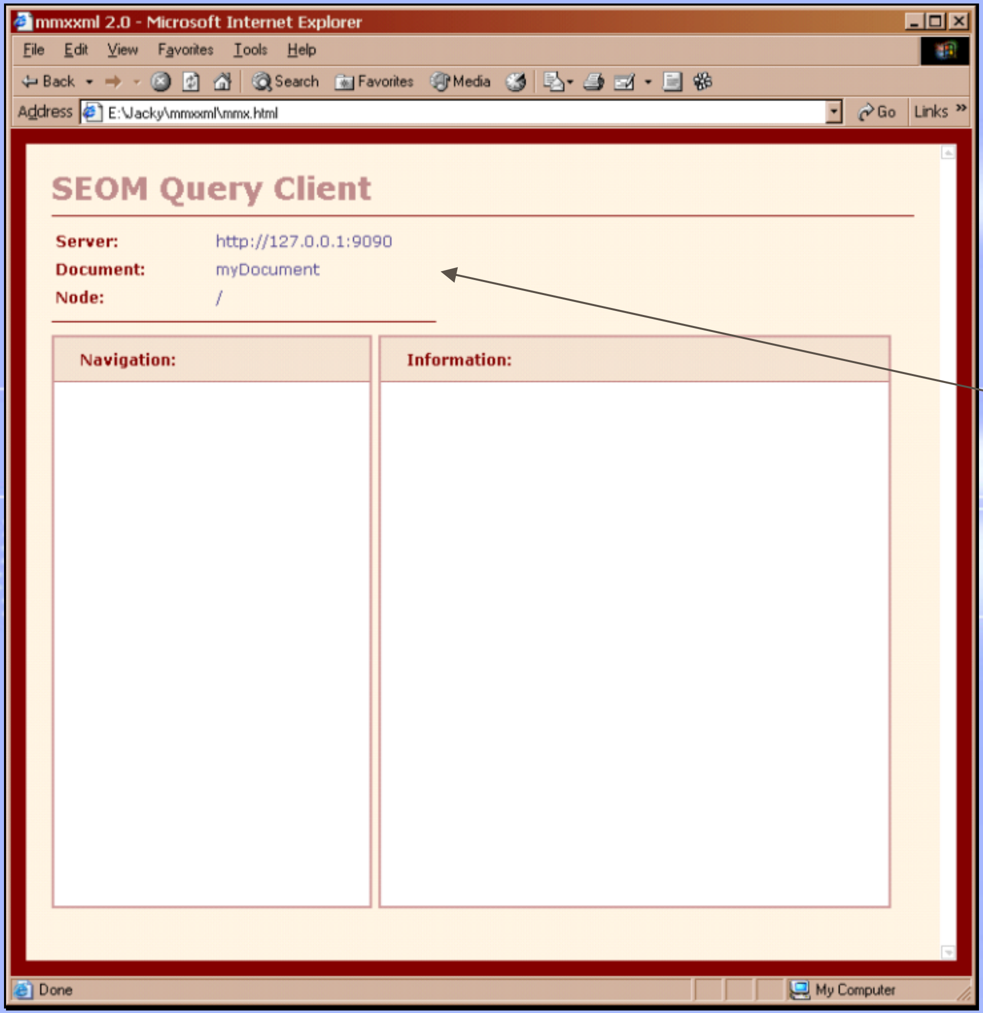
System Design (Server)



System Design (Client)



Interface


mmxxml 2.0 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address E:\Jacky\mmxml\mmx.html
Go Links
Done
My Computer

SEOM Query Client

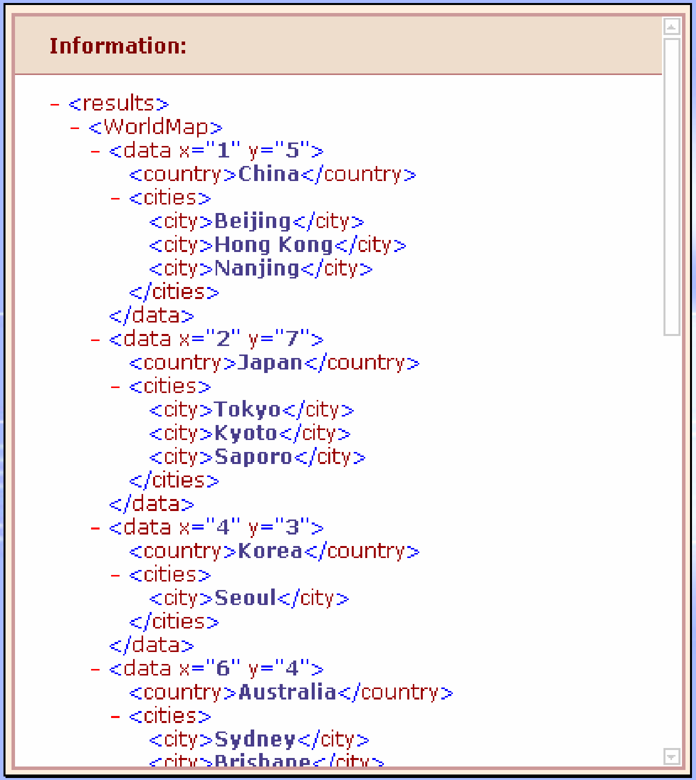
Server:	http://127.0.0.1:9090
Document:	myDocument
Node:	/

Navigation:

Information:

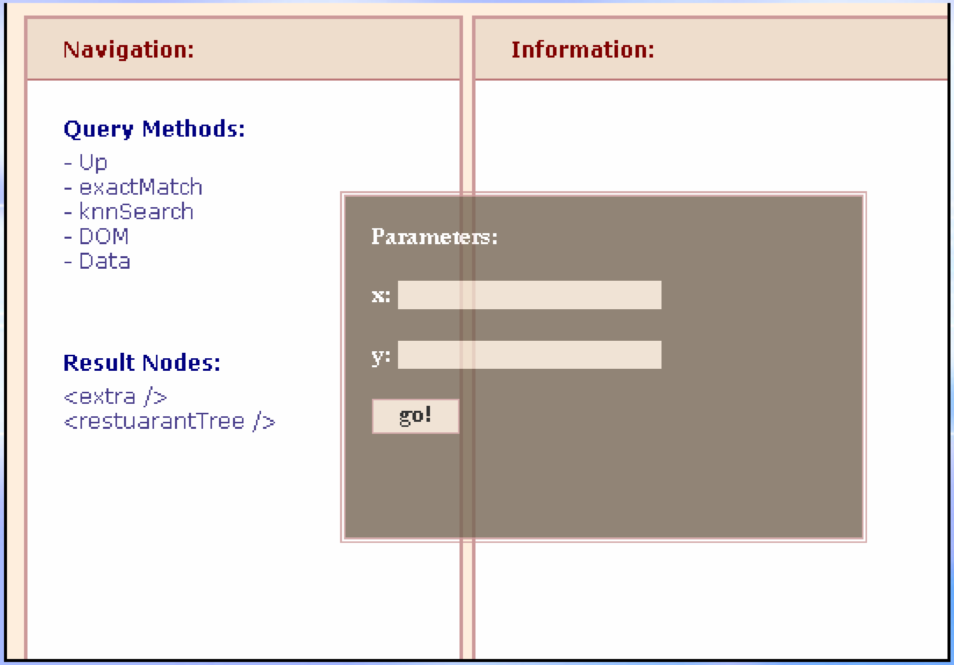
Server:	http://127.0.0.1:9090
Document:	myDocument
Node:	/

Interface



Information:

```
- <results>
  - <WorldMap>
    - <data x="1" y="5">
        <country>China</country>
      - <cities>
          <city>Beijing</city>
          <city>Hong Kong</city>
          <city>Nanjing</city>
        </cities>
      </data>
    - <data x="2" y="7">
        <country>Japan</country>
      - <cities>
          <city>Tokyo</city>
          <city>Kyoto</city>
          <city>Saporo</city>
        </cities>
      </data>
    - <data x="4" y="3">
        <country>Korea</country>
      - <cities>
          <city>Seoul</city>
        </cities>
      </data>
    - <data x="6" y="4">
        <country>Australia</country>
      - <cities>
          <city>Sydney</city>
          <city>Brisbane</city>
```



Navigation:

Query Methods:

- Up
- exactMatch
- knnSearch
- DOM
- Data

Result Nodes:

<extra />
<restuarantTree />

Information:

Parameters:

x:

y:

Evaluation

Discussions - Pros

- Separates logical data representation from physical data representation, thus hides the unnecessary details from the programmers
- Data can be validated semantically during object instantiation
- Flexible internal data structure implementation for SElement allows better optimization on data processing

Discussion – Pros(2)

- Coincides with object-oriented programming paradigm: object encapsulation, modular development, and reusable software components
- Flattens legacy data structures into XML, which is text-editable, easy to transport and process by different systems
- Facilitates interoperability through the use of schema
- Inherits the DOM interface, and can use other technologies built for DOM

Discussion - Cons

- The size of XML file is often larger than legacy data file
- Overhead in parsing is longer
- Each structure model needs additional implementation effort
- Proprietary schema specification may not attract people to use

Means of Enhancement

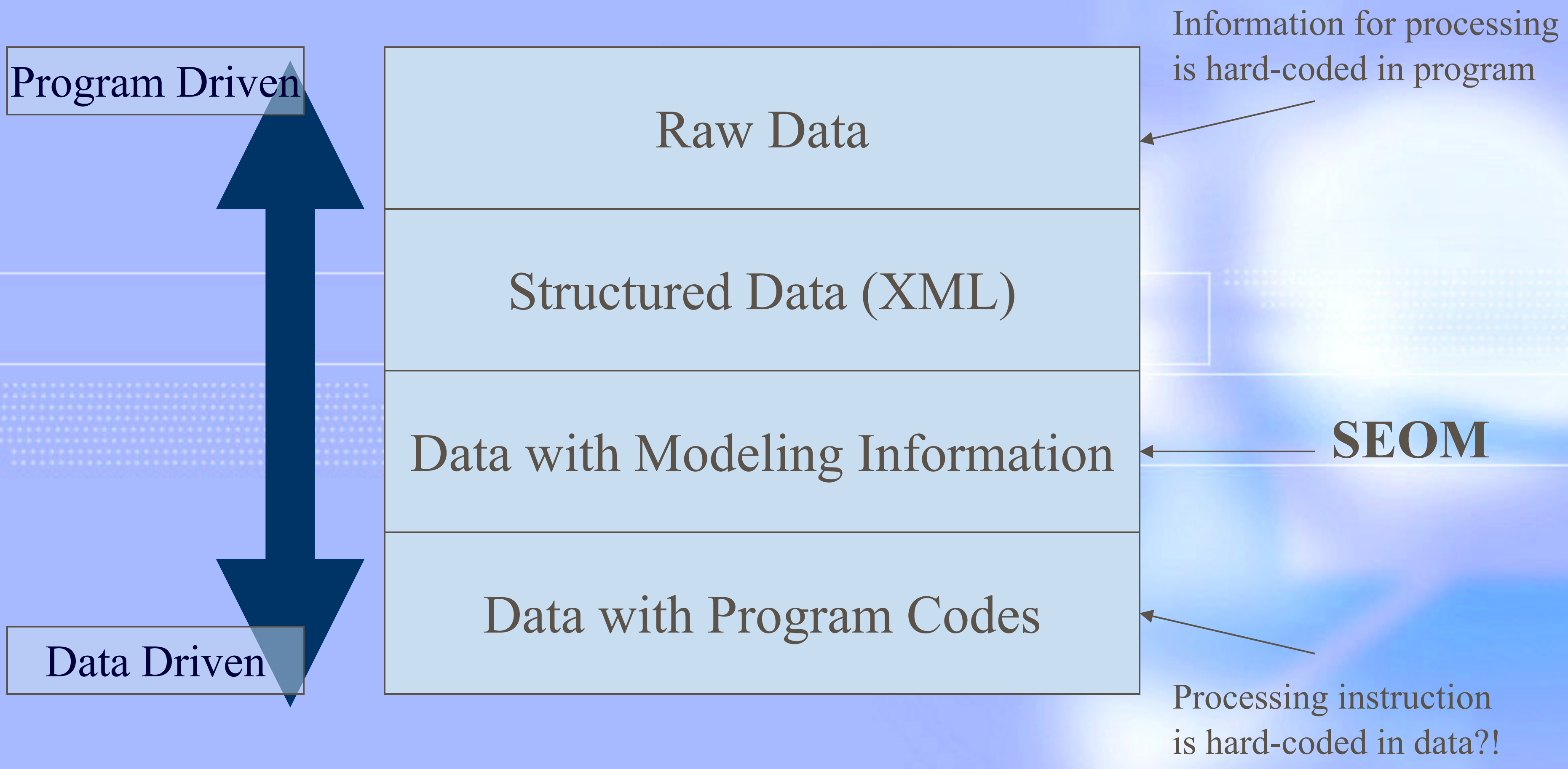
- Include other manipulation methods such as inserting data, removing data, serialize to XML etc.
- Include referencing mechanisms to define graph relation, which is more general and expressive than the hierarchical relationship

Conclusion

- An object model combining the features of DOM and data binding technology
- A schema for mapping physical XML data into Java classes that implement logical entities
- A framework to facilitate marshaling and unmarshaling between XML data and the data objects
- A mechanism to support querying SElements by exchanging XML wrapper messages
- An extension of the XPath to support filtering nodes using the query function in SElement
- A web-based XML query system using SEOM has been implemented to demonstrate our work

Q&A

Program Driven vs. Data Driven


Program Driven
Data Driven
Raw Data
Structured Data (XML)
Data with Modeling Information
Data with Program Codes
Information for processing is hard-coded in program
SEOM
Processing instruction is hard-coded in data?!