



# Systematic document generation from XML Schema

Antonia Bertolino, Jinghua Gao, Eda Marchetti, Andrea Polini

name.surname@isti.cnr.it

Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo"  
(ISTI-CNR), Pisa





# Agenda

- XML and XML Schema
- Motivating XML-based Partition Testing (XPT)
- Category Partition(CP) & Mapping from CP to XPT
- XPT Methodology
- TAXI Tool
- Applications
- Conclusion

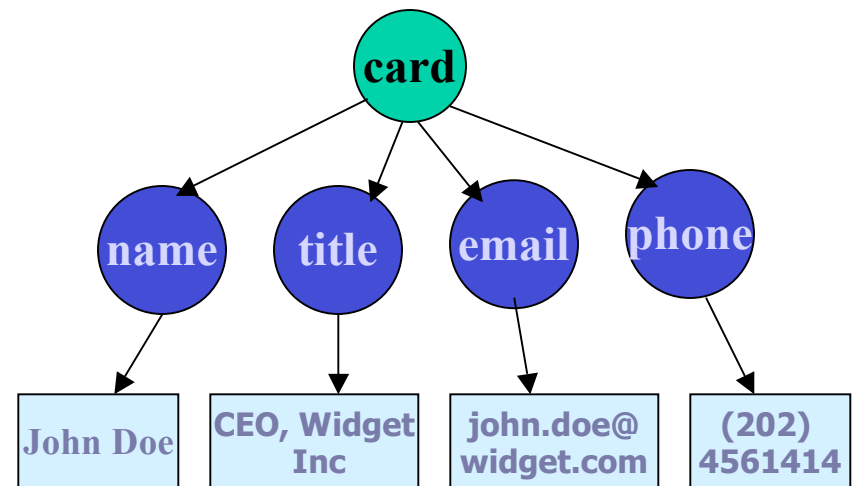




# The eXtensible Markup Language(XML)

- The eXtensible Markup Language (XML) is a Markup Language which is today a de-facto standard to store information and data.
- XML documents are tree structured documents in which data are formatted/organised using tags

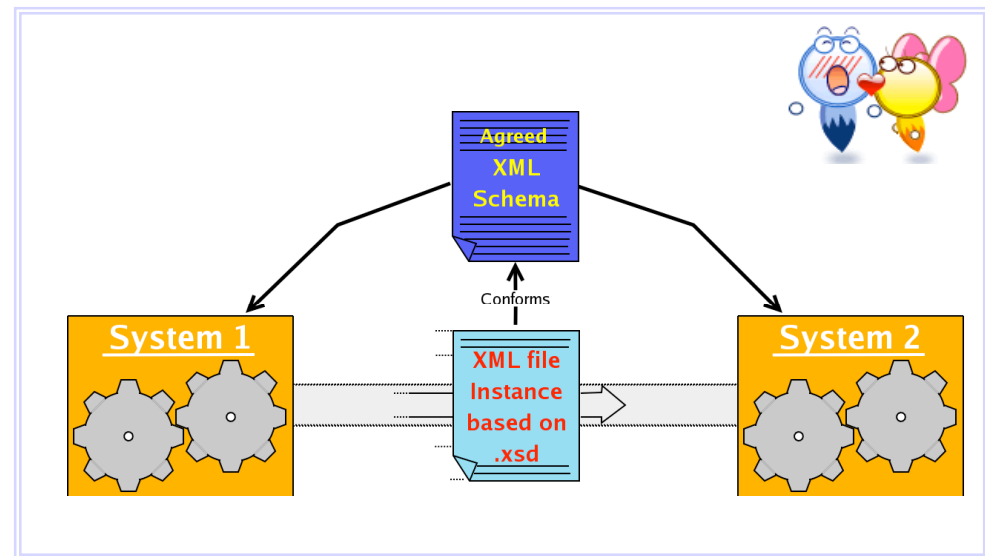
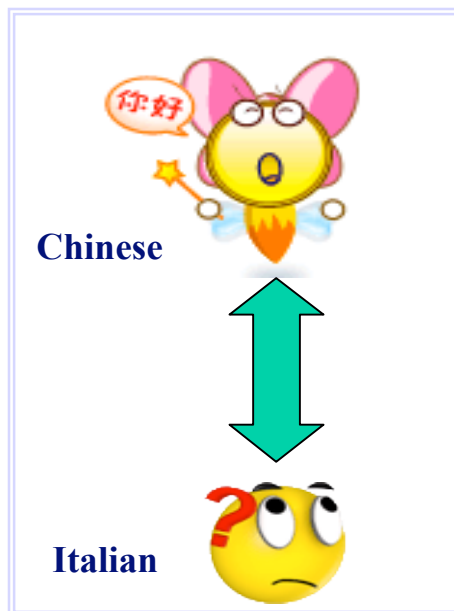
```
<?xml version="1.0"
encoding="ISO88591"?>
<card>
<name>John Doe</name>
<title>CEO, Widget Inc.</title>
<email>john.doe@widget.com</email>
<phone>(202) 4561414</phone>
</card>
```





# XML & XML Schema

- XML Schema provides a means for defining the structure and content of XML documents
- In the open networked world, XML Schema support interoperability between independently developed applications





# Automatic XML-Based Testing and Benchmarking

What we would like  
to achieve



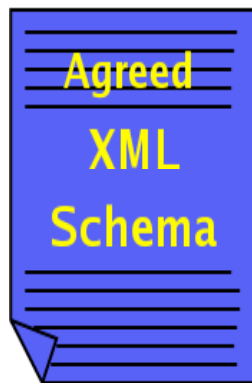
*Generally not  
Finite set!!*





# Automatic XML-Based Testing and Benchmarking

## EASY WAY



*Not Structured set, difficult to derive properties!!*

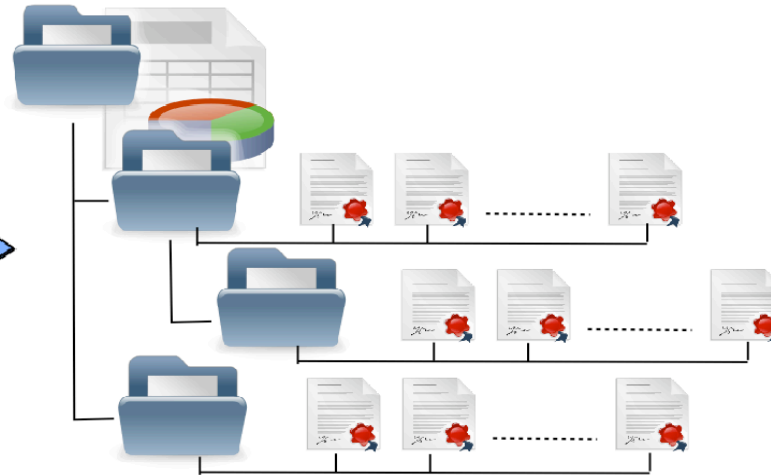
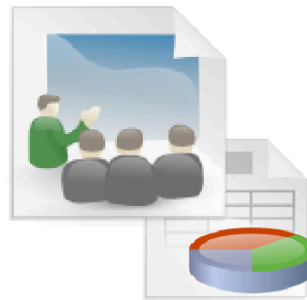
Some tools like that exist: XMLSpy, sunXMLGenerator, ...





# Our proposal: A Systematic Automatic Approach

## XML-based Partition Testing XPT



*A structured set of instances, with  
some predefined properties!*

The approach has been inspired at-large by the well-known **Category Partition** methodology for systematic semi-automated test generation ...

..or, you can think of it as **grammar-based generation**, on the XSD syntax, although we have also introduced practical rules












# Mapping CP to XPT

**CP**

**XPT**

 Analyze Specifications	→	Preprocessor
 Identify Functional Units	→	Identify Sub-Schema Sets
 Partition Categories	→	Identify Types
 Select Choices	→	Partition Values and Structures
 Determine Constraints	→	Determine "valid/invalid" constraints
 Generate Test Specification	→	Generate Intermediate Instances
 Generate Test Cases	→	Generate Final Instances

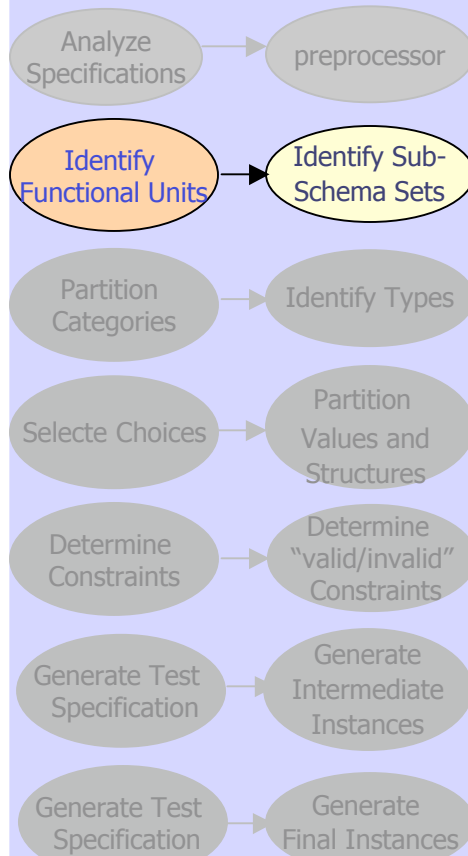




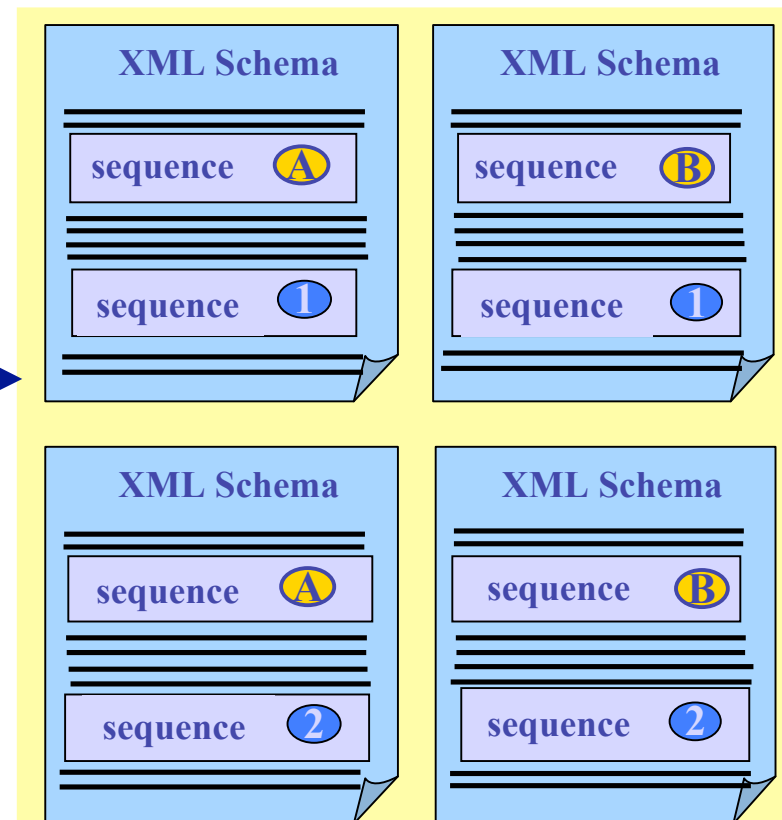
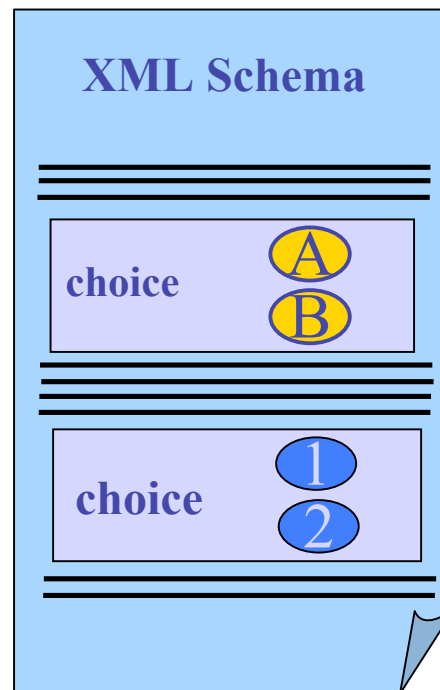


# Identification of Sub-Schema Sets

## Mapping from CP to XPT



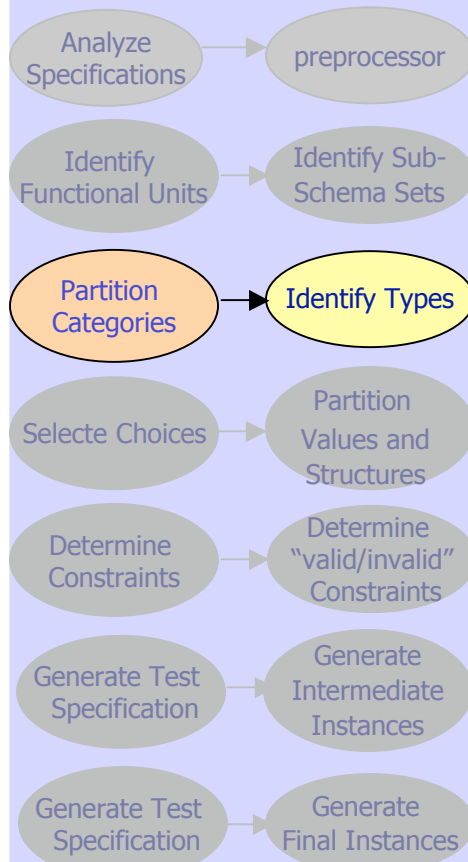
- **<choice> elements partition the XML Schema into distinct set corresponding to the CP functional units**





# Identification of Types

## Mapping from CP to XPT



The CP categories in XPT correspond to the occurrence and types of XML elements.

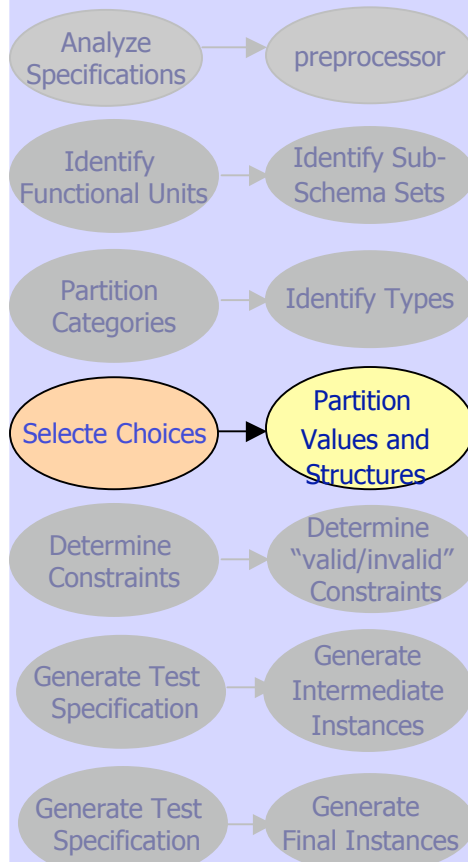
➤ EX: String, sequence, all





# Partition of Values and Structures

## Mapping from CP to XPT



## ➤ Values of the elements:

- Element attributes: fixed, default ...
- Restrictions: "minInclusive", "maxInclusive", "minExclusive", "maxExclusive", "minLength", "maxLength"

## ➤ Information of the structure of the final instances

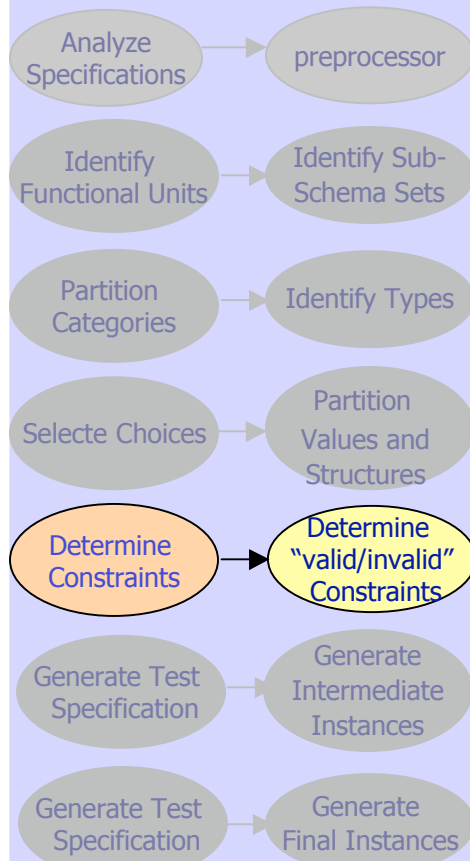
- Element: "minOccurs", "maxOccurs"
- Attribute: "use"





# Constraints of “valid/invalid”

## Mapping from CP to XPT



- Two types of constraints can be identified
  - **Valid:** values in choices conform to the specification of the XML Schema
  - **Invalid:** values in choices do not conform to the declaration of XML Schema.





# Example of “valid/invalid” constraints

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="purchaseorder">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="name" type="nameType"/>
      <xs:element name="productName" type="xs:string"/>
      <xs:element name="price" type="xs:decimal" minOccurs="0"/>
      <xs:element name="address" type="addressType" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:complexType name="nameType">
  <xs:all>
    <xs:element name="firstName" type="xs:string"/>
    <xs:element name="lastName" type="xs:string"/>
  </xs:all>
</xs:complexType>
<xs:complexType name="addressType">
  <xs:choice>
    <xs:element name="USaddress" type="xs:string" maxOccurs="unbounded"/>
    <xs:element name="EUAddress" type="xs:string"/>
  </xs:choice>
</xs:complexType>
</xs:schema>
```

## Sequence:

The same sequence of element as the specification of XML Schema [Valid]

The sequence of element is different from the XML Schema specification [Invalid]

## Numeric:

Any digitals conform to the specification of XML Schema [Valid]

Any digitals do not conform to the specification of XML schema. [Invalid]

## String:

Any strings conform to the specification of XML Schema [Valid]

Any strings do not conform to the specification of XML schema. [Invalid]

## Occurrence:

Occurrence value  $\in (-\infty, \text{minOccurs})$  [Invalid]

Occurrence value  $\in [\text{minOccurs}, \text{maxOccurs}]$  [Valid]

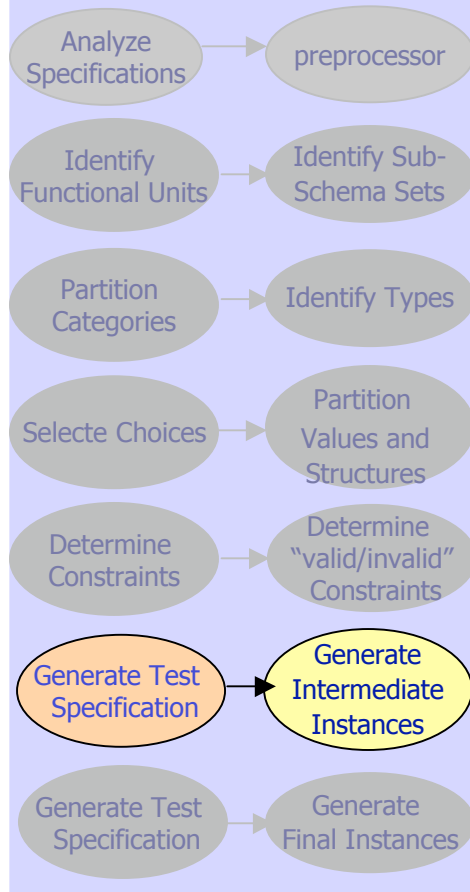
Occurrence value  $\in (\text{maxOccurs}, \infty)$  [Invalid]





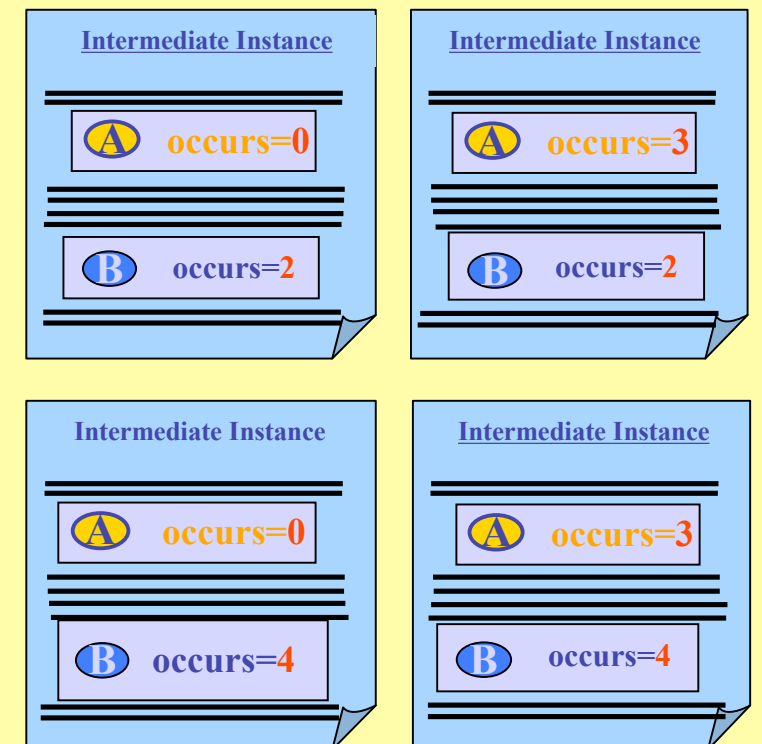
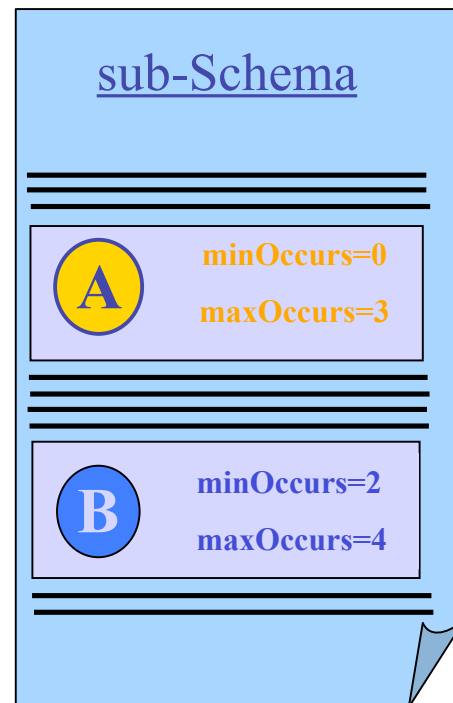
# Intermediate Instances

## Mapping from CP to XPT



➤ Generate intermediate instance by combining the values of "minOccurs" and "maxOccurs".

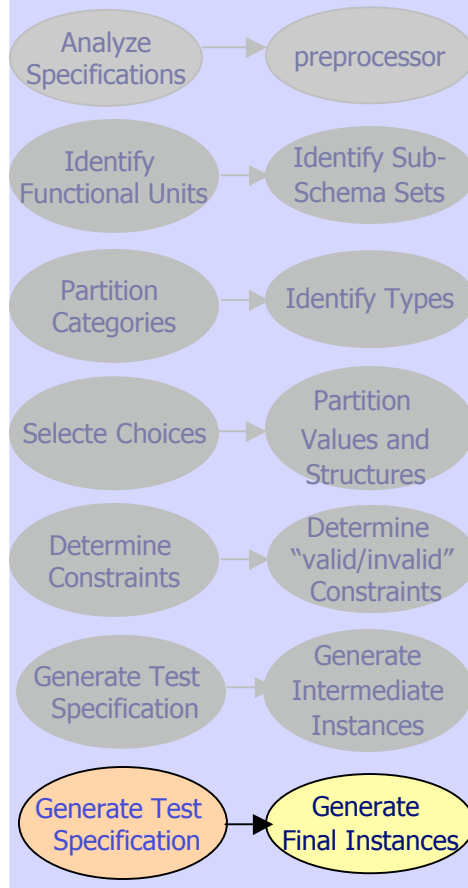
➤ We apply the **conventional Boundary Condition** test approach to reduce the combinations





# Instance Derivation

## Mapping from CP to XPT



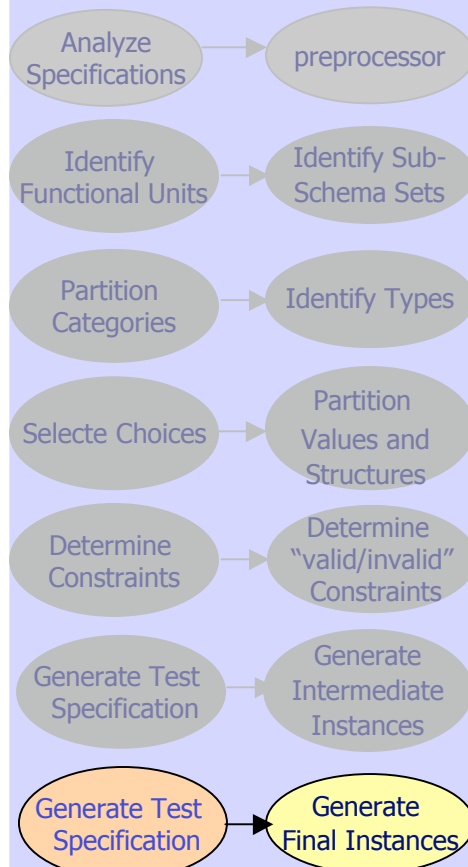
- The set of final instances is generated by giving the proper value to each element.
- The values are selected from the choices according to the **restrictions** expressed in the XML Schema.



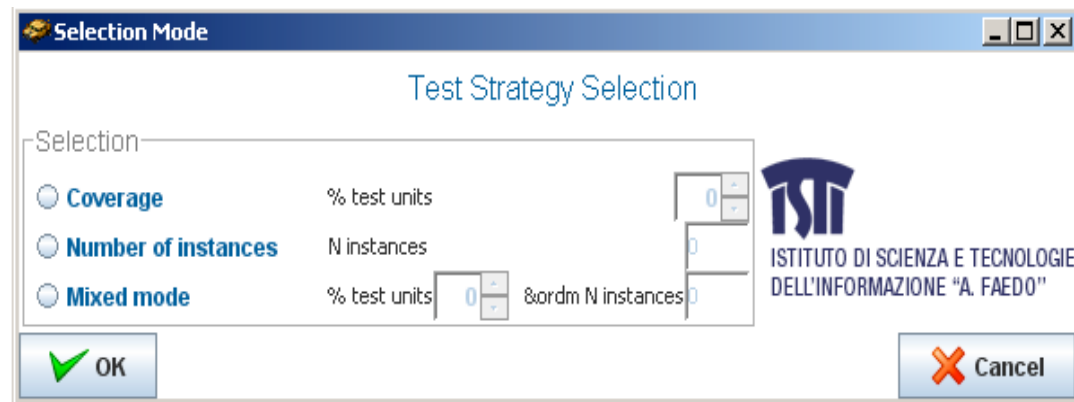


# Instance Derivation(2)

## Mapping from CP to XPT



- The problem of CP method: **Too many generations!**
- Our solution:
  - Apply **Pair-wise testing** during the occurrence generation
  - **Weighted Test Strategies**







# Main Interface of TAXI

Open Save Expand all Collapse all Start modify Stop modify Reset

ISTITUTO DI SCIENZA E DELL'INFORMAZIONE "A"

Tree Nodes	Weight Value	minOccurs	maxOccurs
xs:restriction			
xs:pattern			
xs:element: expense-item			
xs:complexType			
xs:sequence			
xs:element			
xs:element			
xs:choice		0	
xs:element	0.1666		
xs:element: Lodging	0.1666		
xs:complexType			
xs:sequence			
xs:element: Name		0	
xs:element			
xs:element	0.1666		
xs:element	0.1666		
xs:element: Entertainment	0.1666		
xs:complexType			
xs:sequence			
xs:element: Client-		0	
xs:element: Misc	0.1666		
xs:complexType			
xs:attribute: mistype			
xs:simpleType			
xs:restriction			
xs:enumerat			
xs:enumerat			
xs:enumerat			

Automatic generation

Show instances

**LEGENDA**

- : document root
- : Node containing a choice in its subtrees.
- : Generic node.
- : Choice child node
- : Choice child node containing a choice in its subtrees.
- : Choice node.
- : Choice node containing a choice in its subtrees.
- : Schema's Element children list.
- : Schema's Element children list containing a choice in its subtrees.

Info Refresh Quit





# TAXI



- The mapping from the CP to the XML Schema Partition Testing has been partially implemented in a proof-of-concept tool called **TAXI : Testing by Automatically generated XML Instances**
- TAXI includes four components
  - **Schema Analyzer (XSA)**
    - Expands and preprocesses the XML Schema,
    - Prepares the intermediate instance frames
    - Provides a set of final instances
  - **Test Strategy Selector (TSS)**
    - Implements a set of test strategies.
    - Manages the weight assignment for the elements in the identified functional units
  - **Values Storage (VS)**
    - Manages a database for occurrences and values assignment
  - **User Interface (UI)**





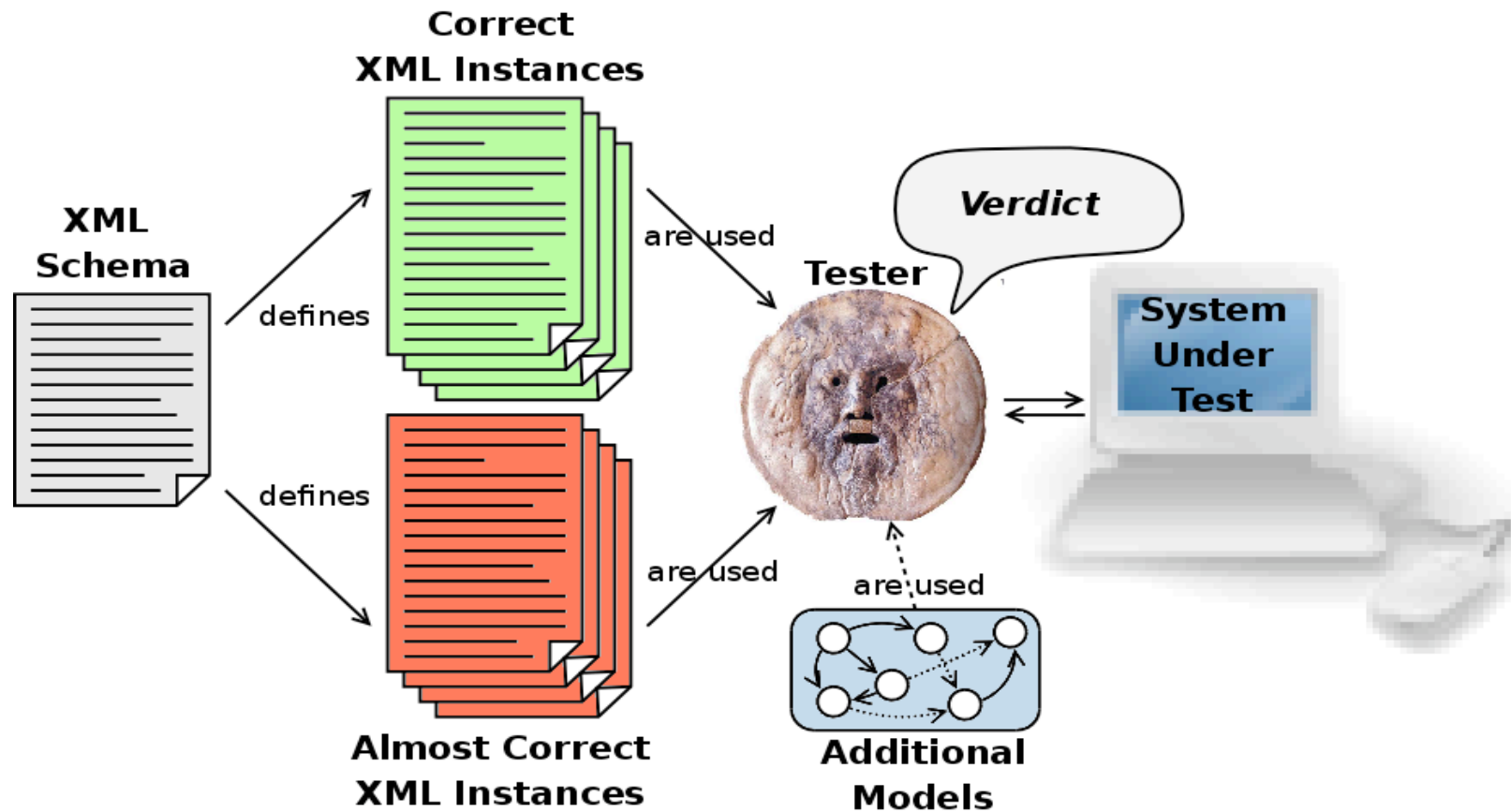
# Potential Applications

- For validating database management systems
  - automatically generate valid XML instances for populating database in a systematic
  - evaluate the performance and the quality of the associated management systems
- For testing the inter-operability between applications and for enabling the correct interactions among the interfaces used by remote components in distributed systems.
  - Automatic and controlled generation of valid and invalid instances enables the automated testing of I/O behavior
- For verifying the proper communication protocols between web-services.
  - SOAP-based interaction between services can be reconducted to the corresponding XML Schemas



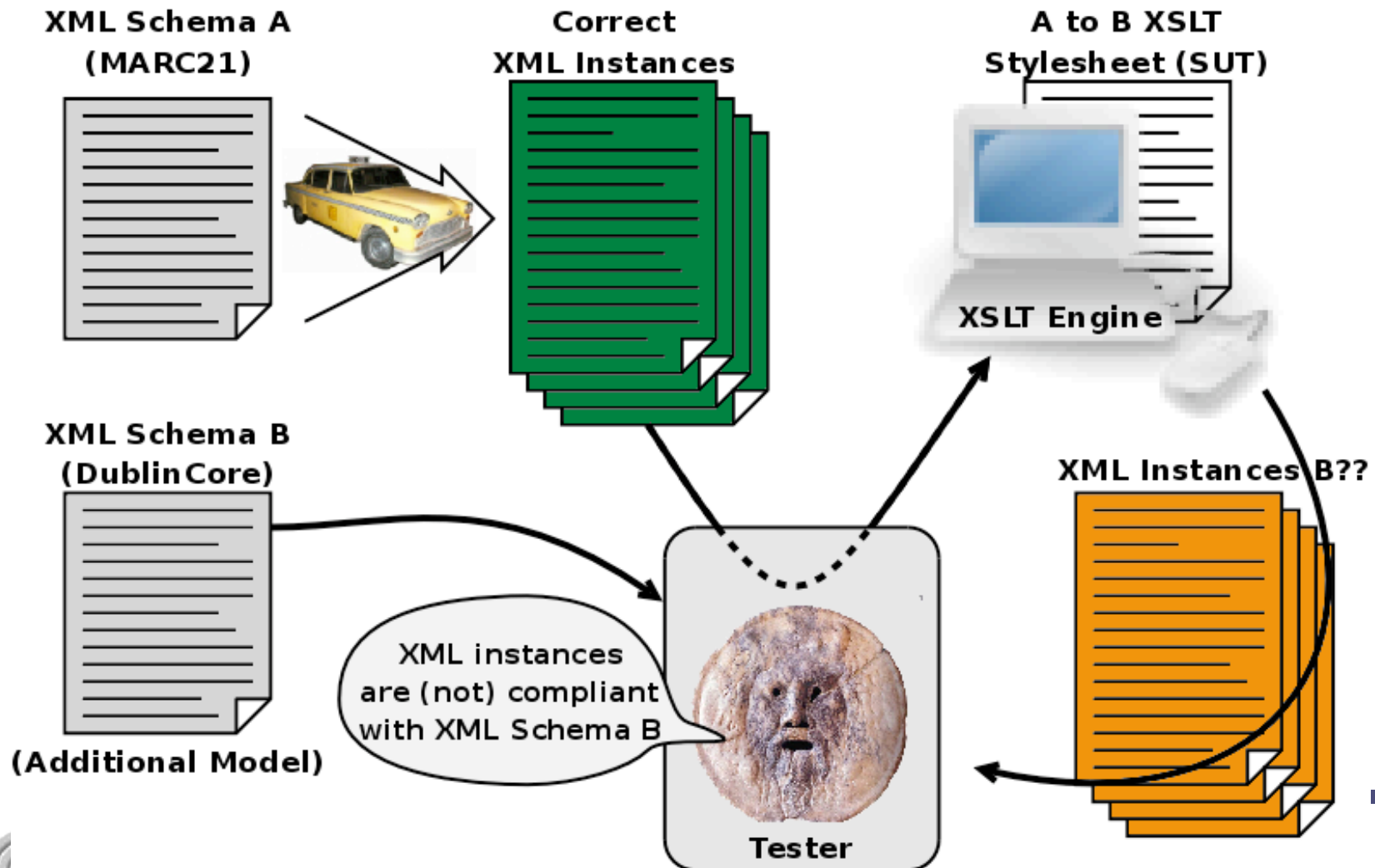


# Black-box testing





# 100% automatic XSLT testing





## Conclusions

- TAXI tool can automatically derive a set of instances that systematically covers a XSD
- It can be applied for interoperability validation, database benchmarking, black-box testing, ...

### Future work

- Invalid instance generation: Robustness testing
- Tool refinement
- Experimental validation





# Thank you!

To get TAXI, or for joint experimentation

- Beta Version of TAXI on line at <http://labse.isti.cnr.it/>

Or

- send an email to [antonia.bertolino@isti.cnr.it](mailto:antonia.bertolino@isti.cnr.it)

