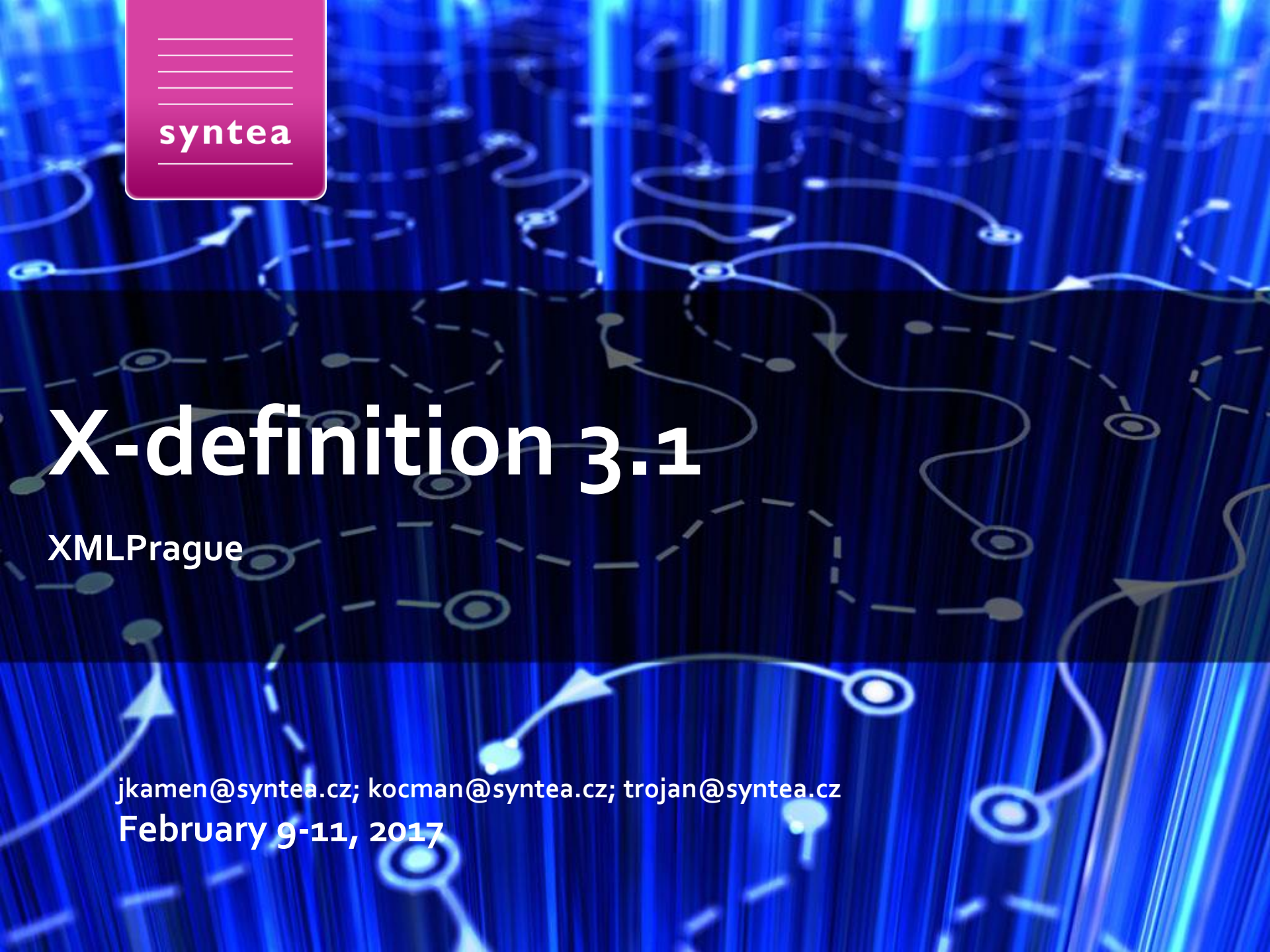


X-definition 3.1

XMLPrague

jkamen@syntea.cz; kocman@syntea.cz; trojan@syntea.cz

February 9-11, 2017

XDefinition 2.0 -> X-definition 3.1

- Improvement of interconnection with external environment: (declaration Java methods, the access the external objects, processing of errors etc.).
- More possibilities of processing and/or construction of XML data (connection to database etc).
- Implementation of all XML schema types including all facets and restrictions and large of set of implemented validation methods (including EBNF). Usage of external Java methods.
- Stream mode (processing of large data).
- Possibility to declare values as unique in given scope of the XML tree and declare references to them.
- Generation of X-components: the Java source code representing XML structures described in X-definitions.
- Implementation of XQuery (eg. version with Saxonia library) and different database tools.
- Processing of JSON data (experimental alpha version).
- http://xdef.syntea.cz/tutorial/en/index.html

Let’s have XML data

```
<inventory date="2017-01-21">;
   <book isbn="123456789" published="2017">
      <title>The X-definitions 3.1</title>
      <author>Jiří Kamenický</author>
      <author>Jindřich Kocman</author>
      <author>Václav Trojan</author>
   </book>
    …
   <book isbn="987654321">
      <title>The Bible</title>
   </book>
</inventory>
```

X-definition: the model of element

```
<xd:def xmlns:xd="http://www.syntea.cz/xdef/3.1"
        xd:root ="inventory"
        xd:name ="Books">
   <inventory date ="date()">
      <book xd:script ="+"
         isbn      ="int()"
         published ="? gYear()">
         <title> string() </title>
         <author xd:script="*"> string() </author>
      </book>
   </inventory>
</xd:def>
```

Complete X-definition

```
<xd:def xmlns:xd="http://www.syntea.cz/xdef/3.1" name="Example" root="Inventory">
  <xd:declaration>
    void message(String s) {
        outln(s);
    }
    int count = 0;
  </xd:declaration>
  <Inventory xd:script="init message('Created ' + now());
    finally if (count > 0) message('Processed ' + count + ' books')">
    <Book xd:script="+; onAbsence message('No books!'); finally count++;"
      isbn="int(0, 9999999999);" published="? gYear();">
    <Title> string(); </Title>
    <Author xd:script="*"> string(); </Author>
    </Book>
  </Inventory>
</xd:def>
```

Execution of XML data with previous X-definition

Input data:

```
<Inventory>
  <Book isbn="1234567890" published="2017">
    <Title>X-definition 3.1</Title>
    <Author>Jiří Kamenický</Author>
    <Author>Jindřich Kocman</Author>
    <Author>Václav Trojan</Author>
  </Book>
  <Book isbn="5678901234">
    <Title>The Bible</Title>
  </Book>
</Inventory>
```

System.out :

```
Created 2017-02-10T12:38:32.523+01:00
Processed 2 books
```

(you can try it on http://xdef.syntea.cz/tutorial/en/example/Example006.html)

Large data 1: ignore some items

```
<inventory date="date()">
   <book xd:script="+"
      isbn      ="int()"
      published="ignore gYear">
      <title> string() </title>
      <author xd:script="ignore"> string() </author>
   </book>
</inventory>
```

Large data 1a: ignore “other” items

```
<inventory xd:script="option ingnoreOther" date="date()">
   <book xd:script="+"
      isbn     ="int()" >
      > <!-- published="ignore gYear“ > -->
      <title> string() </title>
      <!-- author xd:script="ignore"> string() </author -->
   </book>
</inventory>
```

Large data 2: remove items after processing

```
<inventory date = "date()">
   <book xd:script = "+; forget"
      isbn      = "  int()"
      published = "? gYear">
      <title> string() </title>
      <author xd:script = "*"> string() </author>
   </book>
</inventory>
```

Large data 3: stream mode

The stream is connected to X-definition from the external environment.

```
<inventory date = "date()">
   <book xd:script = "+; forget"
      isbn  = "int()"
      published = "? gYear">
      <title> string() </title>
      <author xd:script = "*"> string() </author>
   </book>
</inventory>
```

JSON and X-definition

```
{"inventory": [
   {"book":
      {"isbn": 123456789, "title": "X-definition 3.1", "edited": 2017,
      "author": ["Jiří Kamenický", "Jindřich Kocman", "Václav Trojan"]}
   },
   {"book":
      {"isbn": 987654321, "title": "The Bible"}
   }
]}
```

Rules of conversion of JSON to XML

- **Names:** If the name is a XML name do nothing:

  "books" -> books

- empty string "" -> _u0_
- Characters which are not allowed in XML name it's converted to the sequence of _ux_ where x is hexadecimal reprezentation of the UTF code value:

  "foo and bar" -> foo_u20_and_u20_bar

- **Values:** "jvalue" (string, number, boolean, null or jstring, jnumber, jboolean, jnull)
- **Array:** <js:array>
- **Map**: attributes or <js:map>
- (MapExtension: auxiliary item if a part of map is represented in XML as attributes and the other part as child elements, but not array)

X-definition of JSON

```
<inventory xmlns:js="http://www.syntea.cz/json/1.0">
   <js:array>
   <book  xd:script="+"
      edited="jnumber()"
      isbn  ="jnumber()"
      title ="jstring()">
      <js:array>
        <author xd:script="*"> jstring() </author>
      </js:array>
   </book>
 </js:array>
</inventory>
```

X-definition of JSON without redundant arrays

```
<inventory>
   <book  xd:script="+"
      edited="jnumber()"
      isbn  ="jnumber()"
      title ="jstring()">
      <author xd:script="*"> jstring() </author>
   </book>
</inventory>
```

X-definition and database (1)

```
<xd:declaration>
   String url = "jdbc:derby://localhost:1527/sample;";
   String user = "app";
   external String password;
   Service service = new Service("jdbc", url, user, password);
   Statement statement = service.prepareStatement(
      "SELECT TITLE,ISBN FROM MYTEST.TITLE ORDER BY TITLE ASC");
</xd:declaration>

<Inventory xd:script="">
   <Book xd:script="occurs *; create statement.query()" isbn="int()">
      <Title> string();</Title>
   </Book>
</Inventory>
```

X-definition and database (2)

In the processing (validation) mode are usually used external methods eg. to check values of code lists in a database:

```
<xd:def …
    methods="boolean myproject.Utils.checkTab(XXData, String);">
  <inventory>
    <!-- Note argument XXData is set to the method checkTab authomatically. -->
    <book xd:script="occurs *; isbn="checkTab('ISBN') ">
     …
    </book>
  </inventory>
  …
package mytest;
public class Utils {
  public static boolean checkTab(XXData xd, String name) {
    return isValid(xd.getTextValue(), name);
  }
```

See example of using database:
http://xdef.syntea.cz/tutorial/en/userdoc/DBExample.pdf

EBNF grammar

Let's finally demostrate power of X-definition on example with EBNF grammar:

```
<xd:def xmlns:xd='http://www.syntea.cz/xdef/3.1‘ xd:root = 'Expression'>
  <xd:BNFGrammar name = "G">
    S ::= [#9#10#13 ]+ /*white spaces*/
    number ::= [0-9]+ ("." [0-9]+)? (("E"|"e")("+"|"-")[0-9]+)?
    identifier ::= [a-zA-Z] [a-zA-Z0-9]*
    value ::= "-"? S? (number | identifier | "(" S? expr S? ")")
    mul ::= value (S? ("*" | "/") S? mul)?
    add ::= mul (S? ("+" | "-") S? add)?
    expr ::= add
  </xd:BNFGrammar>

  <Expression> G.rule('expr'); </Expression>
</xd:def>
```

You can try it: http://xdef.syntea.cz/tutorial/examples/BNF.html
And look to X-definition of X-definitions:
http://xdef.syntea.cz/tutorial/examples/XDefinitionOfXDefinition.pdf

What is X-component

- X-components are extensions of X-definitions
- The main purpose is to simplify Marshalling/Unmarshalling when processing XML using the X-definitions
- X-component is a standard java class that implements interface cz.syntea.xc.XComponent. It may be declared as an extension of a superclass and/or to implement interfaces.
- The structure of X-component corresponds to an element from the source X-definition where attributes are class fields and subelements are other X-components
- Each element in the source X-definition constructs one X-component as its image in Java

Generating X-component

- X-components are generated automatically
- Commnads for generating are part of the source package of X-definitions
- Every change of the source X-definition propagates to the X-components

Example

```
<Insurer
            company     ="string()">
   <Contract xd:script="*"
            ContractID  ="int()"
            ValidFrom   ="datetime('yyyyMMdd')"
   />
</Insurer>

<xd:component>
   %class cz.syntea.xmlprague.example.Insurer %link Insurer#Insurer
</xd:component>
```

X-component - Example

```
package cz.syntea.xmlprague.example;
public class Insurer implements cz.syntea.xc.XComponent{
  public String getCompany() {return _Company;}
  public java.util.List<Insurer.Contract> listOfContract() {return _Contract;}
  public void setCompany(String x) {_Company = x;}
  public void addContract(Insurer.Contract x) {
    if (x!=null) _Contract.add(x);
  }
  // Constructors and implementation of cz.syntea.xc.XComponent
  public static class Contract implements cz.syntea.xc.XComponent{
    public Long getContractID() {return _ContractID;}
    public cz.syntea.common.sys.SDatetime getValidFrom() {return _ValidFrom;}
    public void setContractID(Long x) {_ContractID = x;}
    public void setValidFrom(cz.syntea.common.sys.SDatetime x) {_ValidFrom = x;}
    public void setValidFrom(java.util.Date x) {
      _ValidFrom=x==null ? Null : new cz.syntea.common.sys.SDatetime (x);
    }
  // Constructors and implementation of cz.syntea.xc.XComponent
  }
}
```

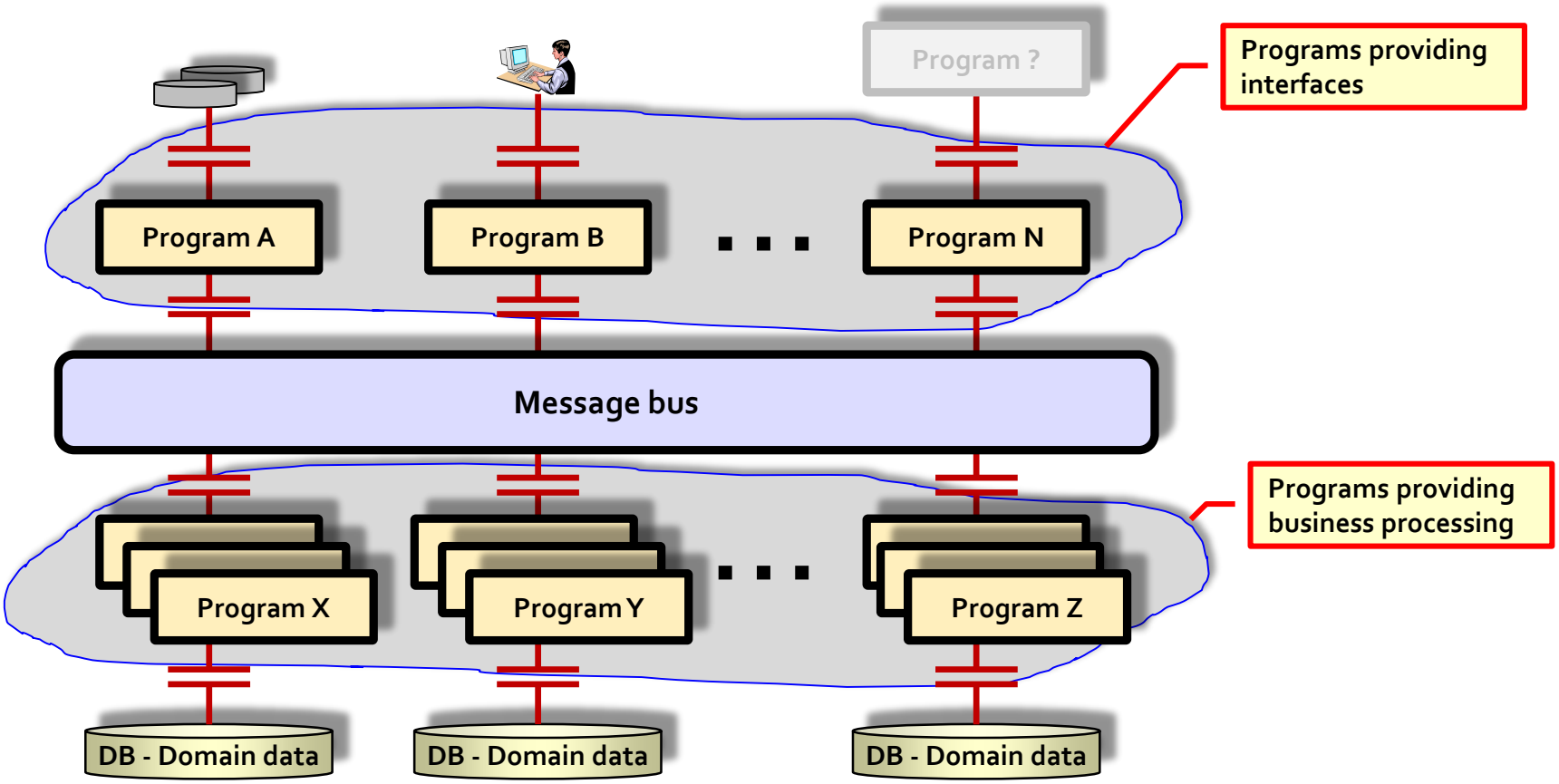
Generating X-component is flexible

- X-component can be descendant of an arbitrary class
- Names of getters and setters can be changed
- Getters and setters can be binded to the superclass instead of generating them. This superclass can provide some additional logic.
- Interface can also be generated.
- This interface contains only getters and setters.

X-component – Example of an interface

```
package cz.syntea.xmlprague.example;

public interface Contract extends cz.syntea.xc.XComponent {

  public Long getContractID();
  public void setContractID(Long x);
  public String xposOfContractID();

  public cz.syntea.common.sys.SDatetime getValidFrom();
  public void setValidFrom(cz.syntea.common.sys.SDatetime x);
  public void setValidFrom(java.util.Date x);
  public String xposOfValidFrom();

}
```

The use of X-definition



The use of X-definition