

Re Write POGO using openArchitectureWare Technology

- Pogo History
- OpenArchitectureWare
technology
- Generated code
- Project status

Re Write POGO using openArchitectureWare Technology

Pogo History

- Pogo project has started in january 2000

First specifications:

Generate a TANGO device class.

Able to re-load after simple modification by programmer.

No xml like file to store information.

Java and C++ generated code must be very closed

Implementation in 2 packages:

Device Model class and generation/load classes

Using String methods to parse code.

Graphic User Interface classes

Re Write POGO using openArchitectureWare Technology

Pogo History

- Tango/Pogo main additional features:
 - Java and C++ api diverged
 - The attribute model changed for C++
 - The inheritance (in C++) must be managed,
 - A 3rd language (python) has been added.

Tango/Pogo minor additional features:

IDL change (today: 4-C++ , 3-Python and 2-Java)

New Types

Polling

Event push

.....

Re Write POGO using openArchitectureWare Technology

Pogo History

- Tango community growing up:
 - 211 device classes on sourceforge.
 - 146 device classes on ESRF repository.
 - Many device classes in each institute.

The backward and universal compatibility is more and more difficult to be maintained with in house parsing.

We would like to re-write it with an advanced parsing tool.

Re Write POGO using openArchitectureWare Technology

openArchitectureWare

- openArchitectureWare (oAW) is a modular MDA/MDD generator framework implemented in Java(TM).
- It supports parsing of arbitrary models, and a language family to check and transform models as well as generate code based on them.
- Supporting editors are based on the Eclipse platform.
- OAW has strong support for EMF (Eclipse Modeling Framework) based models but can work with other models, too (e.g. UML2, XML or simple JavaBeans)
- At the core there is a workflow engine allowing the definition of generator/transformation workflows.
- A number of prebuilt workflow components can be used for reading and instantiating models, checking them for constraint violations, transforming them into other models and then finally, for generating code.

<http://www.openarchitectureware.org/>

Re Write POGO using openArchitectureWare Technology

openArchitectureWare

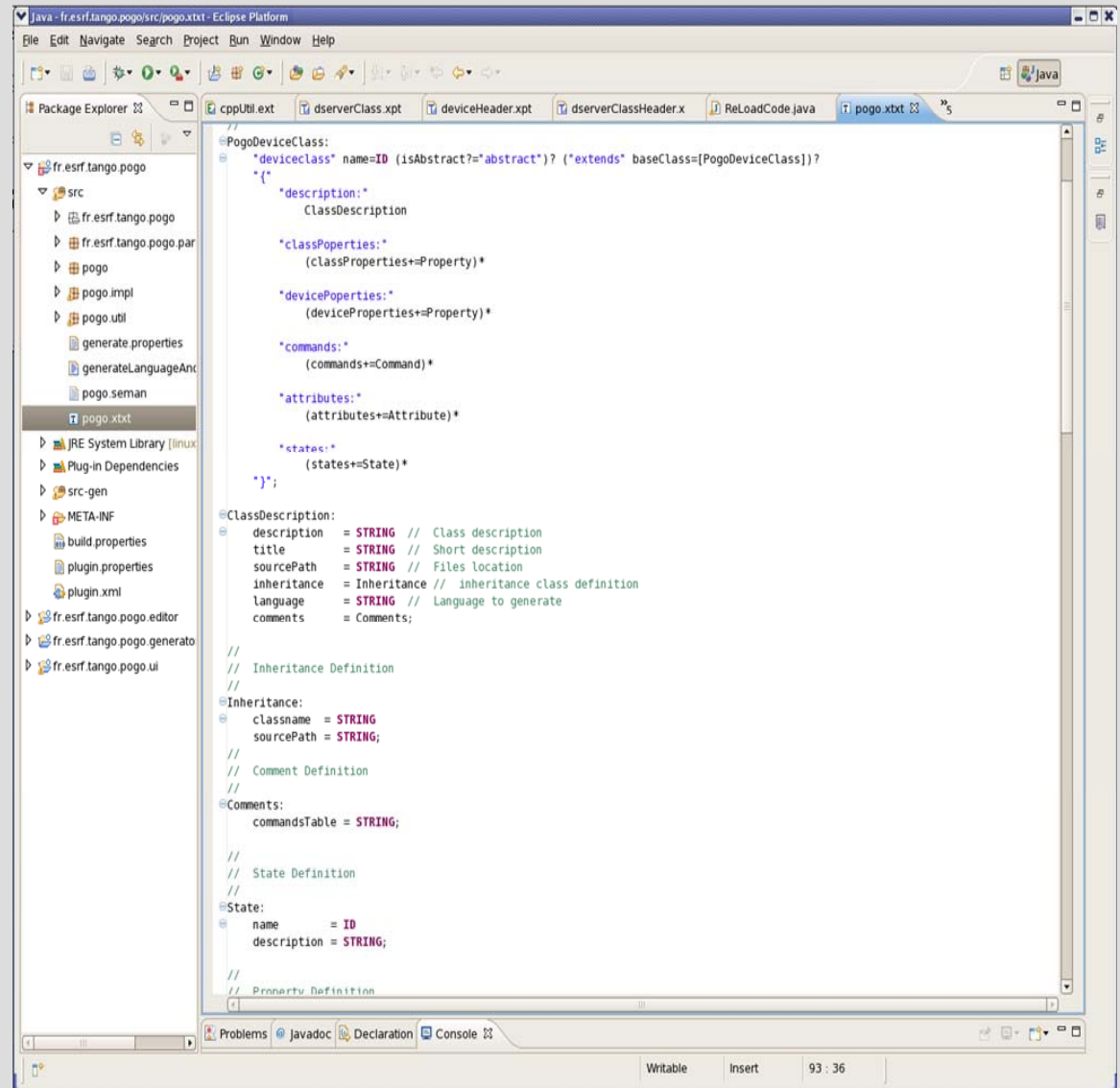
Defining your grammar (Xtext):

A Tango device class is defined by:

- Class description:
 - Description
 - Inheritance
 - Language
 - -----
- A list of class properties
- A list of device properties
- A list of commands
- A list of attributes
- A list of states
- -----

A command is defined by:

- A name
- A description
- An Input argument
 - Type
 - Description
- -----



The screenshot shows the Eclipse IDE with the Xtext grammar for POGO device classes. The Package Explorer on the left shows the project structure for 'fr.esrf.tango.pogo'. The main editor displays the Xtext grammar file 'pogo.xtext' with the following content:

```
grammar PogoDeviceClass;
'deviceclass' name=ID (isAbstract?='abstract'? (*extends* baseClass={PogoDeviceClass})?
  '{'
    'description:'
      ClassDescription
    'classProperties:'
      (classProperties+=Property)*
    'deviceProperties:'
      (deviceProperties+=Property)*
    'commands:'
      (commands+=Command)*
    'attributes:'
      (attributes+=Attribute)*
    'state:'
      (states+=State)*
  '}';

@ClassDescription:
description = STRING // Class description
title = STRING // Short description
sourcePath = STRING // Files location
inheritance = Inheritance // inheritance class definition
language = STRING // Language to generate
comments = Comments;

// Inheritance Definition
//
@Inheritance:
classname = STRING
sourcePath = STRING;

// Comment Definition
//
@Comments:
commandsTable = STRING;

// State Definition
//
@State:
name = ID
description = STRING;

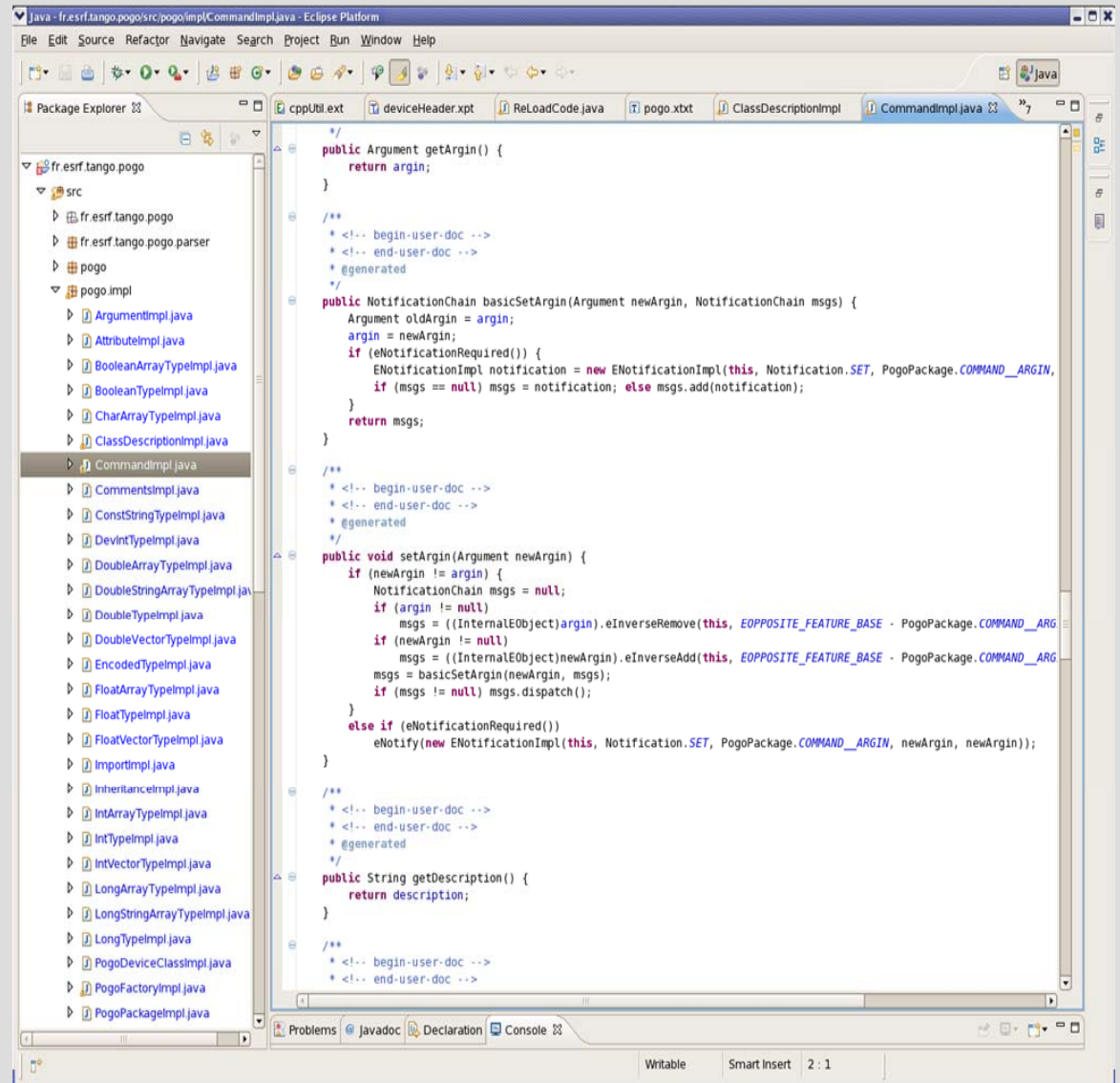
// Property Definition
```

Re Write POGO using openArchitectureWare Technology

openArchitectureWare

Generate the
Eclipse Modeling
Framework
(EMF) model

Generate
Java API classes
for getters and
setters



The screenshot shows the Eclipse IDE interface. On the left, the Package Explorer displays a project structure for 'fr.esrf.tango.pogo'. The 'pogo.impl' package is expanded, showing various implementation classes, with 'CommandImpl.java' selected. The main editor window displays the code for 'CommandImpl.java'. The code includes a package declaration, imports, and several methods: 'getArgin()', 'basicSetArgin()', 'setArgin()', and 'getDescription()'. The code is annotated with Javadoc comments and generated markers like '@generated'.

```
Java - fr.esrf.tango.pogo/src/pogo/impl/CommandImpl.java - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer
fr.esrf.tango.pogo
  src
    fr.esrf.tango.pogo
    fr.esrf.tango.pogo.parser
    pogo
    pogo.impl
      ArgumentImpl.java
      AttributeImpl.java
      BooleanArrayTypemImpl.java
      BooleanTypemImpl.java
      CharArrayTypemImpl.java
      ClassDescriptionImpl.java
      CommandImpl.java
      CommentsImpl.java
      ConstStringTypemImpl.java
      DevIntTypemImpl.java
      DoubleArrayTypemImpl.java
      DoubleStringArrayTypemImpl.java
      DoubleTypemImpl.java
      DoubleVectorTypemImpl.java
      EncodedTypemImpl.java
      FloatArrayTypemImpl.java
      FloatTypemImpl.java
      FloatVectorTypemImpl.java
      ImportImpl.java
      InheritanceImpl.java
      IntArrayTypemImpl.java
      IntTypemImpl.java
      IntVectorTypemImpl.java
      LongArrayTypemImpl.java
      LongStringArrayTypemImpl.java
      LongTypemImpl.java
      PogoDeviceClassImpl.java
      PogoFactoryImpl.java
      PogoPackagemImpl.java

cppUtil.txt deviceHeader.xpt ReLoadCode.java pogo.xtbt ClassDescriptionImpl CommandImpl.java

public Argument getArgin() {
    return argin;
}

/**
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 * @generated
 */
public NotificationChain basicSetArgin(Argument newArgin, NotificationChain msgs) {
    Argument oldArgin = argin;
    argin = newArgin;
    if (eNotificationRequired()) {
        ENotificationImpl notification = new ENotificationImpl(this, Notification.SET, PogoPackage.COMMAND__ARGIN,
            if (msgs == null) msgs = notification; else msgs.add(notification);
    }
    return msgs;
}

/**
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 * @generated
 */
public void setArgin(Argument newArgin) {
    if (newArgin != argin) {
        NotificationChain msgs = null;
        if (argin != null)
            msgs = ((InternalEObject)argin).eInverseRemove(this, EOPOSITIVE_FEATURE_BASE - PogoPackage.COMMAND__ARG
            if (newArgin != null)
                msgs = ((InternalEObject)newArgin).eInverseAdd(this, EOPOSITIVE_FEATURE_BASE - PogoPackage.COMMAND__ARG
            msgs = basicSetArgin(newArgin, msgs);
            if (msgs != null) msgs.dispatch();
        }
        else if (eNotificationRequired())
            eNotify(new ENotificationImpl(this, Notification.SET, PogoPackage.COMMAND__ARGIN, newArgin, newArgin));
    }
}

/**
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 * @generated
 */
public String getDescription() {
    return description;
}

/**
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 */

```

Re Write POGO using openArchitectureWare Technology

openArchitectureWare

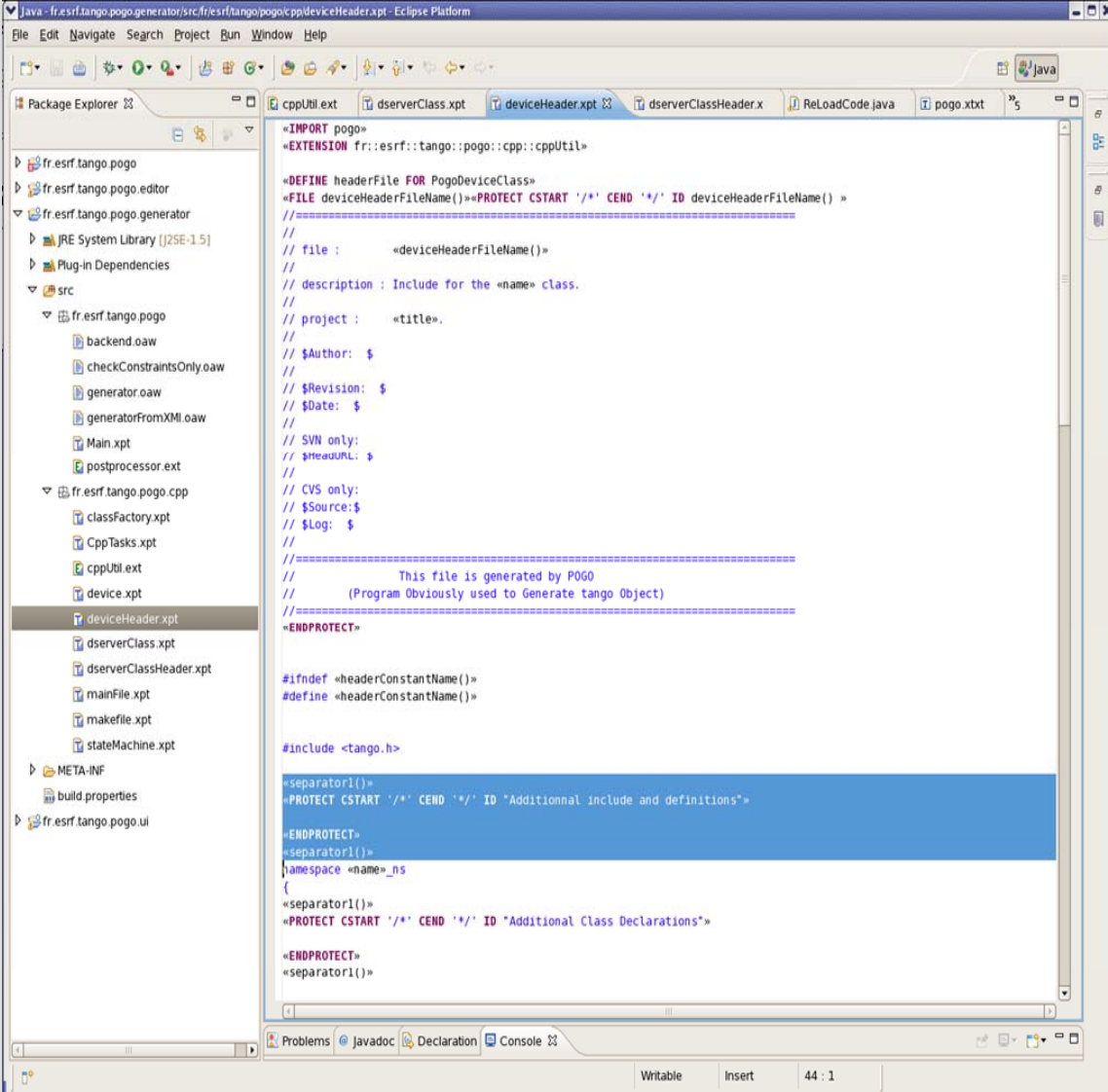
Define

the code templates

with Xpand

And insert protected
areas

e.g. for C++class header



```
«IMPORT pogo»
«EXTENSION fr::esrf::tango::pogo::cpp::cppUtil»

«DEFINE headerFile FOR PogoDeviceClass»
«FILE deviceHeaderFileName()»«PROTECT CSTART '/*' CEND '*/' ID deviceHeaderFileName() »
=====
// file :      «deviceHeaderFileName()»
//
// description : Include for the «name» class.
//
// project :   «title».
//
// $Author: $
//
// $Revision: $
// $Date: $
//
// SVN only:
// $HEADURL: $
//
// CVS only:
// $Source:$
// $Log: $
//
=====
//          This file is generated by POGO
//          (Program Obviously used to Generate tango Object)
=====
«ENDPROTECT»

#ifndef «headerConstantName()»
#define «headerConstantName()»

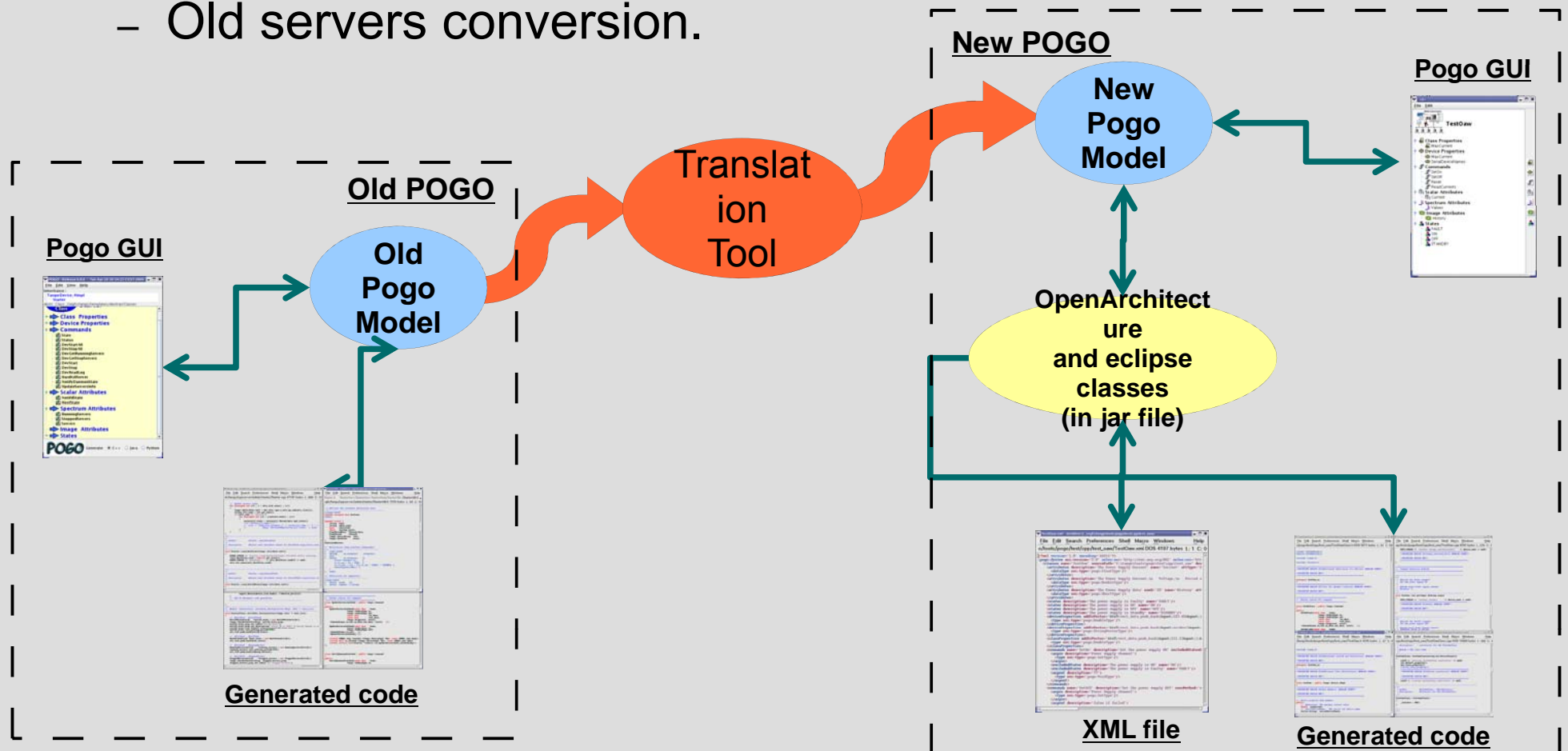
#include <tango.h>

«separator1()»
«PROTECT CSTART '/*' CEND '*/' ID "Additional include and definitions"»
«ENDPROTECT»
«separator1()»
namespace «name»_ns
{
«separator1()»
«PROTECT CSTART '/*' CEND '*/' ID "Additional Class Declarations"»
«ENDPROTECT»
«separator1()»
}
```

Re Write POGO using openArchitectureWare Technology

Generated code

- The generated code will be closed to the old Pogo:
 - Programmers habits.
 - Old servers conversion.



Re Write POGO using openArchitectureWare Technology

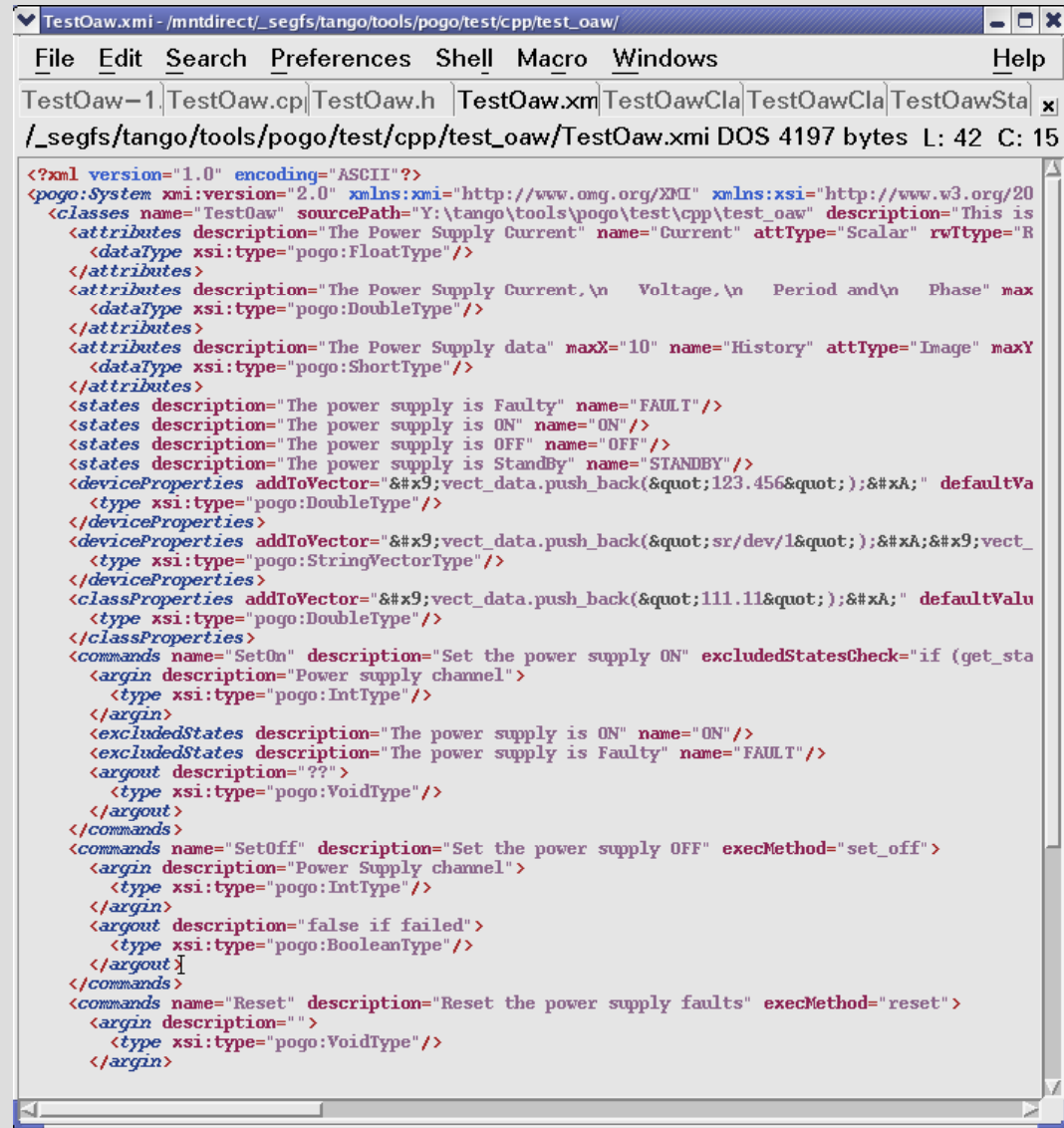
Generated code

- Difficulties are mainly to convert for old servers to new model:
 - More than 500 classes to be converted,
 - Many of them have extra methods, classes, and unexpected things.
How to parse and where insert them ?
 - No way back.

Re Write POGO using openArchitectureWare Technology

Generated code

An XML file
defining
the POGO model
to reload later



```
TestOaw.xmi - /mntdirect/_segfs/tango/tools/pogo/test/cpp/test_oaw/
File Edit Search Preferences Shell Macro Windows Help
TestOaw-1 | TestOaw.cp | TestOaw.h | TestOaw.xml | TestOawCla | TestOawCla | TestOawSta | x
/_segfs/tango/tools/pogo/test/cpp/test_oaw/TestOaw.xmi DOS 4197 bytes L: 42 C: 15
<?xml version="1.0" encoding="ASCII"?>
<pogo:system xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  <classes name="TestOaw" sourcePath="Y:\tango\tools\pogo\test\cpp\test_oaw" description="This is
    <attributes description="The Power Supply Current" name="Current" attType="Scalar" rwType="R
      <dataType xsi:type="pogo:FloatType"/>
    </attributes>
    <attributes description="The Power Supply Current, \n Voltage, \n Period and \n Phase" max
      <dataType xsi:type="pogo:DoubleType"/>
    </attributes>
    <attributes description="The Power Supply data" maxX="10" name="History" attType="Image" maxY
      <dataType xsi:type="pogo:ShortType"/>
    </attributes>
    <states description="The power supply is Faulty" name="FAULT"/>
    <states description="The power supply is ON" name="ON"/>
    <states description="The power supply is OFF" name="OFF"/>
    <states description="The power supply is StandBy" name="STANDBY"/>
    <deviceProperties addToVector="&#x9;vect_data.push_back(&quot;123.456&quot;);&#xA;" defaultVa
      <type xsi:type="pogo:DoubleType"/>
    </deviceProperties>
    <deviceProperties addToVector="&#x9;vect_data.push_back(&quot;sr/dev/l&quot;);&#xA;&#x9;vect_
      <type xsi:type="pogo:StringVectorType"/>
    </deviceProperties>
    <classProperties addToVector="&#x9;vect_data.push_back(&quot;111.11&quot;);&#xA;" defaultValu
      <type xsi:type="pogo:DoubleType"/>
    </classProperties>
    <commands name="SetOn" description="Set the power supply ON" excludedStatesCheck="if (get_sta
      <argin description="Power supply channel">
        <type xsi:type="pogo:IntType"/>
      </argin>
      <excludedStates description="The power supply is ON" name="ON"/>
      <excludedStates description="The power supply is Faulty" name="FAULT"/>
      <argout description="?">
        <type xsi:type="pogo:VoidType"/>
      </argout>
    </commands>
    <commands name="SetOff" description="Set the power supply OFF" execMethod="set_off">
      <argin description="Power Supply channel">
        <type xsi:type="pogo:IntType"/>
      </argin>
      <argout description="false if failed">
        <type xsi:type="pogo:BooleanType"/>
      </argout>
    </commands>
    <commands name="Reset" description="Reset the power supply faults" execMethod="reset">
      <argin description="">
        <type xsi:type="pogo:VoidType"/>
      </argin>
    </commands>
  </classes>
</pogo:system>
```

Re Write POGO using openArchitectureWare Technology

Generated code

Code generated
with
protected areas

```
«IMPORT pogo»
«EXTENSION fr::esrf::tango::pogo::cpp::cppUtil»

«DEFINE headerFile FOR PogoDeviceClass»
«FILE deviceHeaderFileName()»«PROTECT CSTART '/*' CEND '*/' ID deviceHeaderFileName() »
//=====
// file :      «deviceHeaderFileName()»
//
// description : Include for the «name» class.
//
// project :    «title».
//
// $Author: $
//
// $Revision: $
// $Date: $
//
// SVN only:
// $HeadURL: $
//
// CVS only:
// $Source:$
// $Log: $
//
//=====
//
//          This file is generated by POGO
//          (Program Obviously used to Generate tango Object)
//=====
«ENDPROTECT»

#ifdef «headerConstantName()»
#define «headerConstantName()»

#include <tango.h>

«separator1()»
«PROTECT CSTART '/*' CEND '*/' ID "Additional include and definitions"»

«ENDPROTECT»
«separator1()»
namespace «name»_ns
{
«separator1()»
«PROTECT CSTART '/*' CEND '*/' ID "Additional Class Declarations"»

«ENDPROTECT»
«separator1()»

/*PROTECTED REGION ID(TestOaw.h) ENABLED START*/
//=====
//
// file :      TestOaw.h
//
// description : Include for the TestOaw class.
//
// project :    TANGO device server.
//
// $Author: $
//
// $Revision: $
// $Date: $
//
// SVN only:
// $HeadURL: $
//
// CVS only:
// $Source:$
// $Log: $
//
//=====
//
//          This file is generated by POGO
//          (Program Obviously used to Generate tango Object)
//=====
/*PROTECTED REGION END*/

#ifndef TESTOAW_H
#define TESTOAW_H

#include <tango.h>

/*PROTECTED REGION ID(Additional include and definitions) ENABLED START*/
//=====
//
//=====
/*PROTECTED REGION END*/

namespace TestOaw_ns
{
/*PROTECTED REGION ID(Additional Class Declarations) ENABLED START*/
//=====
//
//=====
/*PROTECTED REGION END*/

class TestOaw : public Tango::Device_4Impl
{
/*PROTECTED REGION ID(Data Members) ENABLED START*/
//=====
//
//=====
/*PROTECTED REGION END*/
}
```

Re Write POGO using openArchitectureWare Technology

Project status

- **Done:**
 - ~40% of GUI.
 - ~40% of grammar
 - ~30% of C++ generated code
 - ~30% of convertor (old -> new model)

- **Not started:**
 - Hinheritance.
 - Java code (will be started when JNI project will be more advanced)
 - Python code
 - Html code
 - New features