AGORA 3.0 Progress

Summary

AGORA2XML (April 2011)

This tool converts AGORA 2 output files into AGORA 3 ones. It has been developed in April 2011, and has to main goals:

- Validate the specifications of the XML output files produced by AGORA 3, bridges between clustering and transcription tools.
- Produce test dataset from the old version of AGORA, and make them available to the other members of the team who work on other workpackages.

How to use

1. Go to the directory that contains AGORA 2 « *projet.xml* » and the subdirectory "*Résultats*".
2. Run *"agora2xml.exe projet.xml"*

How it works

The following tasks are executed on the bounce:

1. Read the AGORA 2 project file
2. Crate the directories *"./images"* , "*./alto*", "*./mets*" , *"./tei*"
3. Copy the images used by AGORA in "*./images*" and normalize their names
4. Create and fill ALTO files : **1 ALTO file per image**
5. Create and fill TEI files TEI : **1 TEI file for the whole project**
6. Create and fill METS files: **1 METS file for the whole project**

ALTO, TEI and METS files specification

ALTO files detailed content

Principles

- ALTO **NNNNN.xml** matches **NNNNN.jpg**
- Classification AGORA → classes list → <ParagraphStyle> list in the ALTO file header
- Use of unique and hierarchic ID (DNS-like). This naming convention is not specified in the ALTO norm, yet, it will simplify the cross references between files (TEI, METS, ALTO)
- An ALTO page *<Page ID= "NNNNN">* contains only one *<PrintSpace>*
- A *<PrintSpace>* contains several TextBloxk: *<TextBlock ID="NNNNN.1">*, *<TextBlock ID="NNNNN.2">, etc.* corresponding to the blocks identified in AGORA within the image *NNNNN.jpg* (same block numbering as AGORA)
- A *<TextBlock>* also contains a *STYLEREF* that allow specifying a class regarding AGORA.
  This class is mentioned in the <ParagraphStyle> list of the ALTO file header.
  Example : *<TextBlock ID="00000.248" STYLEREFS="TEXTE" HEIGHT="42" WIDTH="29" HPOS="641" VPOS="203">*
- A <TextBlock> contains several *<TextLine ID="NNNNN.B.1">, <TextLine ID="NNNNN.B.2">*, etc. corresponding to the lines identified in AGORA in the "*blocB.xml*" file.
- A <TextLine> contains several *<String ID="NNNNN.B.L.1.1">, <String ID="NNNNN.B.L.1.2">, …, <String ID="NNNNN.B.L2.1">, <String ID="NNNNN.B.L.2.2">, etc.*
- These *<String>* correspond to the *Connected Components* (CC) that are present in the "*blocB.xml*" file.
  A *<String> CONTENT* (mandatory attribute in ALTO) is set to *"*"* (*i.e.* the CC isn(t recognized)
  Example : *<String ID="00000.248.0.0.0" HEIGHT="11" WIDTH="7" HPOS="641" VPOS="203" CONTENT="*" />*
- ALTO doesn't recognize the word (*i.e.* sequence of characters) structuring, d'où l'identification du mot d'appartenance d'une String grâce à son ID : ID="00001.5.4.3.2" signifie « 2nd String of 3rd word of 4th clock of 5th of the page 00001 »

- The ALTO tag *<SP> ("Space")* isn't exploied yet. Thes tag is often used by OCR software to separate the <String>

Example of a generated ALTO file

```
<?xml version="1.0" encoding="utf-8"?>
<alto xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.loc.gov/standards/alto/ns-v2#">
  <Description>
    <MeasurementUnit>pixel</MeasurementUnit>
    <sourceImageInformation>
      <fileName>00000.jpg</fileName>
      <fileIdentifier fileIdentifierLocation="./images" />
    </sourceImageInformation>
    <OCRProcessing ID="AGORA5.0" />
  </Description>
  <Styles>
    <ParagraphStyle ID="TEXTE" />
    <ParagraphStyle ID="IMAGE" />
    <ParagraphStyle ID="BRUIT" />
    <ParagraphStyle ID="TITRE" />
    <ParagraphStyle ID="IMGTITRE" />
    <ParagraphStyle ID="MD" />
  </Styles>
  <Layout>
    <Page ID="00000" PHYSICAL_IMG_NR="0">
      <PrintSpace HEIGHT="1958" WIDTH="1246" HPOS="3" VPOS="1">
        <TextBlock ID="00000.235" STYLEREFS="TEXTE" HEIGHT="81" WIDTH="85" HPOS="515" VPOS="184">
          <TextLine ID="00000.235.0" HEIGHT="81" WIDTH="85" HPOS="515" VPOS="184">
            <String ID="00000.235.0.0.0" HEIGHT="81" WIDTH="85" HPOS="515" VPOS="184" CONTENT="*" />
          </TextLine>
        </TextBlock>
        <TextBlock ID="00000.236" STYLEREFS="TEXTE" HEIGHT="42" WIDTH="28" HPOS="476" VPOS="194">
          <TextLine ID="00000.236.0" HEIGHT="39" WIDTH="22" HPOS="482" VPOS="197">
            <String ID="00000.236.0.0.0" HEIGHT="8" WIDTH="12" HPOS="483" VPOS="212" CONTENT="*" />
            <String ID="00000.236.0.0.1" HEIGHT="39" WIDTH="22" HPOS="482" VPOS="197" CONTENT="*" />
          </TextLine>
        </TextBlock>
        <TextBlock ID="00000.237" STYLEREFS="TEXTE" HEIGHT="17" WIDTH="11" HPOS="700" VPOS="197">
          <TextLine ID="00000.237.0" HEIGHT="17" WIDTH="11" HPOS="700" VPOS="197">
            <String ID="00000.237.0.0.0" HEIGHT="17" WIDTH="11" HPOS="700" VPOS="197" CONTENT="*" />
          </TextLine>
        </TextBlock>
        <TextBlock ID="00000.238" STYLEREFS="TEXTE" HEIGHT="14" WIDTH="11" HPOS="580" VPOS="204">
          <TextLine ID="00000.238.0" HEIGHT="14" WIDTH="11" HPOS="580" VPOS="204">
            <String ID="00000.238.0.0.0" HEIGHT="14" WIDTH="11" HPOS="580" VPOS="204" CONTENT="*" />
          </TextLine>
        </TextBlock>
        <TextBlock ID="00000.239" STYLEREFS="TEXTE" HEIGHT="23" WIDTH="8" HPOS="423" VPOS="207">
          <TextLine ID="00000.239.0" HEIGHT="23" WIDTH="8" HPOS="423" VPOS="207">
            <String ID="00000.239.0.0.0" HEIGHT="23" WIDTH="8" HPOS="423" VPOS="207" CONTENT="*" />
          </TextLine>
        </TextBlock>
        <TextBlock ID="00000.240" STYLEREFS="TEXTE" HEIGHT="20" WIDTH="24" HPOS="437" VPOS="206">
          <TextLine ID="00000.240.0" HEIGHT="14" WIDTH="23" HPOS="437" VPOS="206">
            <String ID="00000.240.0.0.0" HEIGHT="11" WIDTH="10" HPOS="437" VPOS="206" CONTENT="*" />
            <String ID="00000.240.0.0.1" HEIGHT="11" WIDTH="14" HPOS="446" VPOS="209" CONTENT="*" />
          </TextLine>
          <TextLine ID="00000.240.1" HEIGHT="8" WIDTH="23" HPOS="438" VPOS="218">
            <String ID="00000.240.1.0.0" HEIGHT="8" WIDTH="11" HPOS="438" VPOS="218" CONTENT="*" />
            <String ID="00000.240.1.1.0" HEIGHT="7" WIDTH="8" HPOS="453" VPOS="218" CONTENT="*" />
          </TextLine>
        </TextBlock>
        ...
```

TEI files detailed content

Principles

- The goal is to persist the current transcription of a project
- TEI coding is in accord with the recommendations of the *"manuelTEIrenaissance"* document (July 2009).
- Initially, the TEIL file produced by AGORA will only contain "*". They will be replaced by character recognized using RETRO OCR systems.
- Class structure such as defined by the user in AGORA. (maybe simplistic … is there a better solution?)
- Therefore, a project is segmented in as many *<DIV TYPE="NOM_CLASSE">* as the number of classes created in AGORA

- Each *<DIV TYPE="NOM_CLASSE">* contains as many *<PB ID="ID_PAGE"> (PB* stands for *PageBreak*) as classified images)
- Each page *<PB>* contains as many *<P ID="ID_PARAGRAPHE">* blocks with *NOM_CLASSE* type, identified by AGORA in this page
- Each paragraph *<P>* contains the block lines *<LB ID="ID_LIGNE">*
- Each line contains its characters. Words are separated by spaces.
- The Id of any object matches the ID used in the ALTO file.

Example of a generated TEI file

```
<?xml version="1.0" encoding="utf-8"?>
<tei xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <teiHeader>
    <fileDesc>
      <titleStmt>
        <title>ProjetTest03</title>
      </titleStmt>
    </fileDesc>
  </teiHeader>
  <text>
    <body>
      <div n="0" type="TEXTE">
        <pb ID="00000">
          <p ID="00000.1">
            <lb ID="00000.1.0">** ****** ******** ******* ***** **</lb>
          </p>
          <p ID="00000.2">
            <lb ID="00000.2.0">* **</lb>
          </p>
          <p ID="00000.3">
            <lb ID="00000.3.0">** ***** ********* * ********** ********** * * ****** ** ******* ***** *******
*** * *** * ****</lb>
            <lb ID="00000.3.1">*** *** ******** ** ** * ********* ****** * ********* ** **** ******** *
******* ** ****** **********</lb>
          </p>
          <p ID="00000.4">
            <lb ID="00000.4.0">* ***** **</lb>
          </p>
          <p ID="00000.5">
            <lb ID="00000.5.0">* *** ****</lb>
          </p>
          ...
```

METS files detailed content

Principles

- The goal is to maintain a logical link between all the files of the project.
- It define the **physical structure** of the page
- It contains:
    - The URL of considered image, assigned to a unique ID
    - The URL of ALTO files, assigned to a unique ID
    - The URL of TEI files, assigned to a unique ID
    - Structural map of the project *<structMap TYPE= "PHYSICAL">*
- The selected granularity is the line (as it was define in AGORA)
- For each line, a precise mapping is created between
    - The image (image ID + coordinates)
    - The ALTO file (ALTO file ID + *<TextLine>* ID*)*
    - The transcription (TEI file ID + *<DIV>* ID)

Example of a generated METS file

```
<?xml version="1.0" encoding="utf-8"?>
<mets xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.loc.gov/METS/">
  <metsHdr>
    <agent ID="AGORA5.0" ROLE="CREATOR" />
  </metsHdr>
  <fileSec>
    <fileGrp ID="ALTO">
```

```
      <file ID="ALTO.00000" MIMETYPE="text/xml">
        <FLocat LOCTYPE="URL" d5p1:type="simple" d5p1:href="file://./alto/00000.xml"
xmlns:d5p1="http://www.w3.org/1999/xlink" />
      </file>
      <file ID="ALTO.00001" MIMETYPE="text/xml">
        <FLocat LOCTYPE="URL" d5p1:type="simple" d5p1:href="file://./alto/00001.xml"
xmlns:d5p1="http://www.w3.org/1999/xlink" />
      </file>
      <file ID="ALTO.00002" MIMETYPE="text/xml">
        <FLocat LOCTYPE="URL" d5p1:type="simple" d5p1:href="file://./alto/00002.xml"
xmlns:d5p1="http://www.w3.org/1999/xlink" />
      </file>
      ...
    </fileGrp>
    <fileGrp ID="IMAGE">
      <file ID="IMAGE.00000" MIMETYPE="image/jpeg">
        <FLocat LOCTYPE="URL" d5p1:type="simple" d5p1:href="file://./images/00000.jpg"
xmlns:d5p1="http://www.w3.org/1999/xlink" />
      </file>
      <file ID="IMAGE.00001" MIMETYPE="image/jpeg">
        <FLocat LOCTYPE="URL" d5p1:type="simple" d5p1:href="file://./images/00001.jpg"
xmlns:d5p1="http://www.w3.org/1999/xlink" />
      </file>
      ...
    </fileGrp>
    <fileGrp ID="TEI">
      <file ID="TEI.ProjetTest03" MIMETYPE="text/xml">
        <FLocat LOCTYPE="URL" d5p1:type="simple" d5p1:href="file://./tei/ProjetTest03.xml"
xmlns:d5p1="http://www.w3.org/1999/xlink" />
      </file>
    </fileGrp>
  </fileSec>
  <structMap TYPE="PHYSICAL">
    <div ID="ProjetTest03" TYPE="DOCUMENT">
      <div ID="00007" TYPE="PAGE">
        <div ID="00007.271" TYPE="BLOCK">
          <div ID="00007.271.0" TYPE="LINE">
            <fptr>
              <area FILEID="IMAGE.00007" COORDS="384,138,77,45" />
            </fptr>
            <fptr>
              <area FILEID="ALTO.00007" BEGIN="00007.271.0" />
            </fptr>
            <fptr>
              <area FILEID="TEI.00007" BEGIN="00007.271.0" />
            </fptr>
          </div>
        </div>
        <div ID="00007.272" TYPE="BLOCK">
          <div ID="00007.272.0" TYPE="LINE">
            <fptr>
              <area FILEID="IMAGE.00007" COORDS="807,136,1242,40" />
            </fptr>
            <fptr>
              <area FILEID="ALTO.00007" BEGIN="00007.272.0" />
            </fptr>
            <fptr>
              <area FILEID="TEI.00007" BEGIN="00007.272.0" />
            </fptr>
          </div>
        </div>
        <div ID="00007.273" TYPE="BLOCK">
          <div ID="00007.273.0" TYPE="LINE">
            <fptr>
              <area FILEID="IMAGE.00007" COORDS="384,197,2118,72" />
            </fptr>
            <fptr>
              <area FILEID="ALTO.00007" BEGIN="00007.273.0" />
            </fptr>
            <fptr>
              <area FILEID="TEI.00007" BEGIN="00007.273.0" />
            </fptr>
          </div>
          ...
```

Librairies

This work contains almost 8000 C# codelines, of which 7000 are automatically generated by *xsd.exe* (Microsoft tool to generate C# classes from a XSD scheme) and slightly modified.

C# Library for reading/writing AGORA project and results

- projetagora.cs, projetagora.xsd
- blocagora.cs, blocagora.xsd

C# Library for reading/writing ALTO

- altov20agora.cs
- altov20agora.xsd (correspond to the ALTO v2 official scheme, slightly modified for .NET)

C# Library for reading/writing TEI

- teiagora.cs
- teiagora.xsd (scheme written regarding the specifications available in "*manuelTEIrenaissance*" document)

C# Library for reading/writing METS

- metsagora.cs
- metsagora.xsd (correspond to the METS v1 official scheme, slightly modified for .NET)

AGORA 3

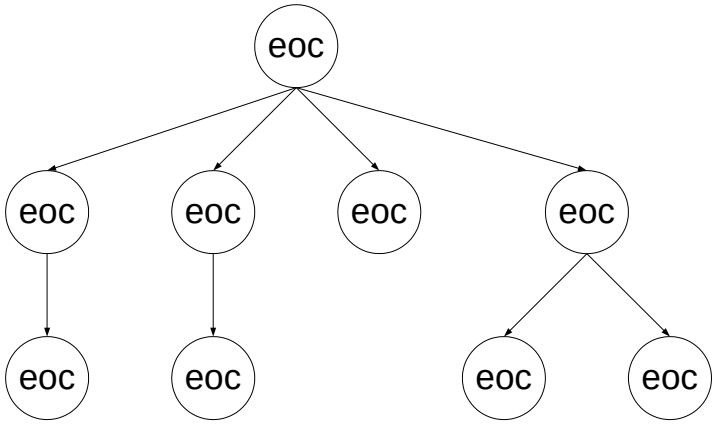
AGORA 3 is currently written with almost 10000 C# codelines, dispatched in 45 classes.

Core of the software – Model

AGORA 3 works on a EOC graph.

Graph = EOC tree

- Node = EOC
  - 2 attributes : type, properties
- oriented arc = relation « is composed of »
  - No attribute
- Initial graph
  - An EOC_DOCUMENT type node
- "Slang" used
  - EOC parent
  - EOC childs



EOC Attributes

Type

1 among N possible types

Ex : *EOC_CC, EOC_LINE, EOC_NOISE, EOC_UserDefined, etc.*

Properties (« points »)

- For each EOC : from 1 to n points
- 1 point = 1 vector of N coordinates

Vector space

- Extensible depending on the requirements
- Has *nbdim* dimensions
- Currently :
  - dimensions 0 et 1 : coordinates X et Y of the Euclidian geometry
  - dimension 2 : value in the grayscale map regarding the horizontal axis
  - dimension 3 : value in the grayscale map regarding the vertical axis

Basic operators on a graph (“ level 0 ” operators)

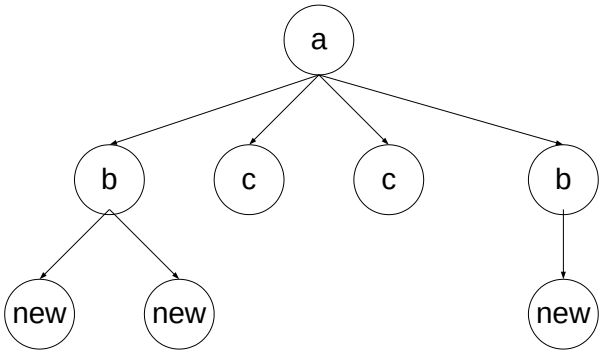
General points

- Operand n°1 of each operator correpond to the root of the subgraph to process
- Recursive behaviour given by the algorithm below.

```
Apply(EOC root)
{
      Foreach (child in root.Childs)
      {
            Apply(child)
      }
      ApplyOperator(root)
}
```

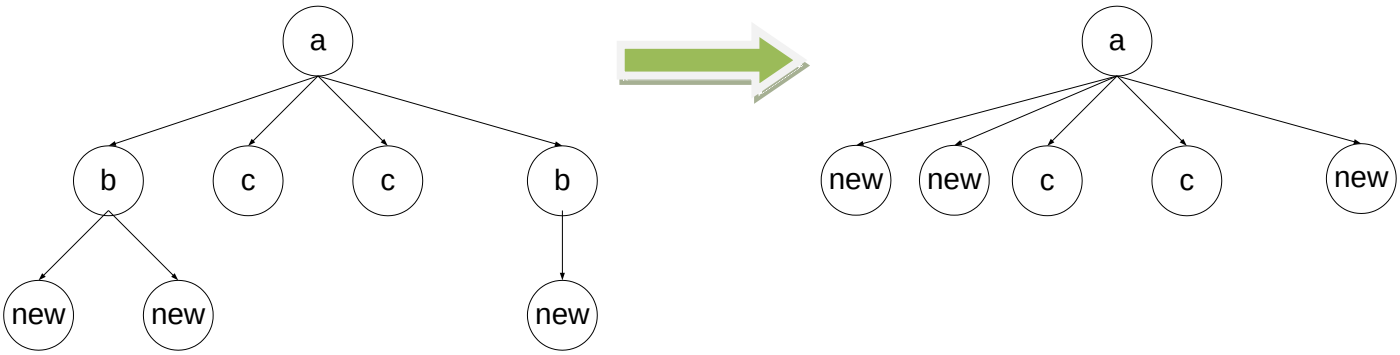
Expand

- 2 operand
    - Root of the subgraph to explore
    - Type of the EOC to expand
- Creation of physical EOC (ex: *CC, Vector, etc.)*
- For each EOC, measure in the vector space (ex : CC = 2 points, vector = 2 points)
- Example : **Expand(Top, « b »)**
- Cf. **OperatorLevel0 ::COperatorExpand.cs** for a example of simple code



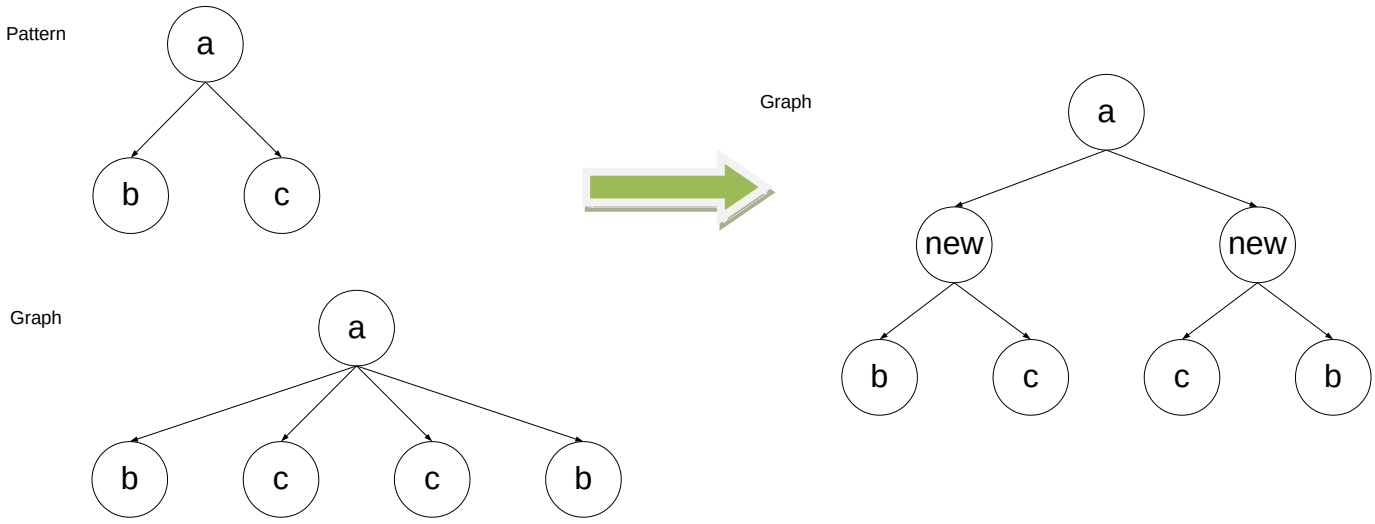
Delete

- Suppression of EOC
- 2 operand
    - Root of the subgraph to explore
    - Type of the EOC to delete
- One parameter = recursive true/false
- Example : **Delete(Top, « b », false)**
- Slightly different version : *DeleteChild*(EOC parent, EOC child to delete, recursive true/false)



Insert

- 2 operand
  - Root of the subgraph to explore
  - Pattern to satisfy
- Pattern
  - EOC 2-level Graph: 1 root and 1..N children
  - Type of the EOC to insert
  - EOC child : has from 1 to n points
  - Score to satisfy
- Basic example of a insert operation



- Algorithm
  1. Find a 1-1 match between a Pattern and a part of the graph
  2. Calculation of the score
  3. If score OK → insert in the graph

InsertGroupElement

It is another version of the Insert operator.

The insertion will be done if the element satisfies the score with one of the element of the group.

- 2 operand
  - Root of the subgraph to explore
  - Pattern to satisfy
- Pattern
  - EOC 2-level Graph: 1 root and 1..N children
  - Type of the EOC to insert
  - EOC Group : has k.N points
  - EOC Element : has N points
  - Score to satisfy

InsertGroupGroup

It is another version of the Insert operator.

The insertion will be done if one of the elements of group 1 satisfies the score with one of the element of group 2.

- 2 operand
  - Root of the subgraph to explore
  - Pattern to satisfy
- Pattern
  - EOC 2-level Graph: 1 root and 1..N children
  - Type of the EOC to insert
  - EOC Group1 : has k1.N points
  - EOC Group2 : has k1.N points
  - Score to satisfy

SetPoint

- Calculation of an EOC points regarding his EOC child points
- Currently several modes:

```
public enum SetPointMode
        {
            No,
            ChildsAll,
            ChildsAverage,
            ChildsMin, ChildsMax,
            ChildsEnveloppe,
            PointsList
        }
```

Score calculation

Basic principles

- When a 1-1 mapping has been found, we gather the points of the matched EOC
    - Y = collection of the n points of the EOC in the graph (nbdim x n matrix)
    - X = colection of the points of the EOC in the pattern (nbdim x n matrix)
- Step 1 : find the best transformation (A,B) such as $\hat{Y}=A.X+B$
- Step 2 : compute the error $||\hat{Y}-Y||$
- Step 3 : compare with the specified max error

Refinements

Constraint imposed on the transformation :

- enableTranslation
    - false : use for pattern about the absolute position of EOC points in the vector space
    - true : use for pattern about the distances between EOC points
- enableScaling
    - allow to avoid scaling issues of the different EOC (for example, used for the W/H ratio of an EOC)
- enableRotation
    - will be used when vector will be manipulated

Projection regarding axis of interest

- Projection matrix de projection to relativize the importance of the dimensions
    - Example : when we work with geometric dimension X/Y of an EOC, we don't care about his position on the grayscale map → we apply a projection matrix with a 0 value in dimensions we want to ignore
- Maximum error to obtain for each projection

Option 1 : pattern contains N EOC « child »

- Simple version (cf §Basics principles)

Option 2 Group/Element: pattern contain 2 EOC « child » n°1 and n°2

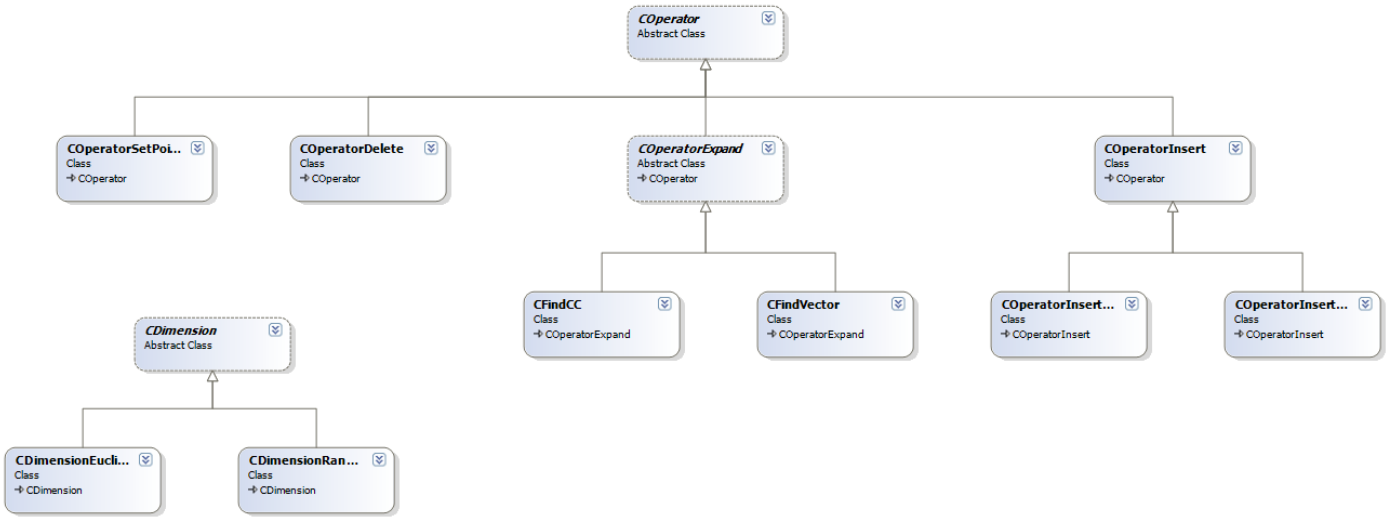
- Usefulness : compare an EOC (n points) to a group of EOC (k x n points)
- Pattern
    - Child n°1 = « Group », child n°2 = « Element »
- Algorithm
    - Y_i (i=1..k) = n points taken in the EOC Group of the graph + n points from the EOC Element (matrix nbdim x 2n)
    - X = points of the 2 EOC of the pattern (matrix nbdim x 2n)

- Then, the same algorithm described above is used with $\hat{Y}$ replaced by $\hat{Y}_i$

Option 3 Group/Group: pattern contains 2 EOC « child » n°1 and n°2

- Usefulness : compare a EOC Group(k x n points) to another EOC Group (k' x n points)
- Pattern
  - Child n°1 = « Group », child n°2 = « Group »
- Algorithm
  - Y_{ij} (i=1..k, j=1..k') = n points taken in EOC Groupe 1 of the graph + n points taken in the EOC Group of the graph (matrix nbdim x 2n)
  - X = points of the 2 EOC of the pattern (matrix nbdim x 2n)
  - Then, the same algorithm described above is used with $\hat{Y}$ replaced by $\hat{Y}_{ij}$

Level 0 operator class diagram



Level 1 Operators

Level 1 operator = mini scenario of level 0 operator. These operators will be directly available from the GUI, therefore, they will constitute the steps of a scenario (heritage of the abstract class Level1 ::CStepOperator)

FusionElementElement

Goal = recognize a specific configuration of 2 EOC. Use case: gather 2 letters to form the beginning of a word.

- Operands : *EOC Parent, EOC ElementL, EOC ElementR, EOC New, Score*
- Fonction : insert *New* id *Score* is satisfy
- Algorithm
  1. Level0 ::Insert(Pattern)

FusionGroupElement

Goal = recognize a specific configuration of 2 EOC, with one of the 2 already belong to a Group. It is the same as inserting 1 EOC in an EOC Group. Use case: add a letter to a neighbor word.

- Operands : *EOC Parent, EOC Group, EOC Element, EOC New, Score*
- Function : insert *New* id *Score* is satisfy
- Algorithm
  1. Recursive call of Level0 ::InsertGroupElement(Pattern)
  2. Graph simplification with Level0 ::Delete()

FusionGroupGroup

Goal = recognize a specific configuration of 2 EOC, with both of them belonging to a Group. It is the same as merging 2 composed EOC. Use case: merge to neighbors words in one.

- Operands : *EOC Parent, EOC Group1, EOC Group2, EOC New, Score*
- Function : insert *New* id *Score* is satisfy
- Algorithm
  1. Recursive call of Level0::InsertGroupGroup(Pattern)
  2. Graph simplification with Level0 ::Delete()

ClassifySize

Goal = recognize an EOC regarding its size, in the desired dimensions.

- Operands : *EOC Parent, EOC Child, EOC New, Score*
- Function : insert *New* id *Score* is satisfy
- Algorithm
  1. Insert(Pattern)

ClassifyForm

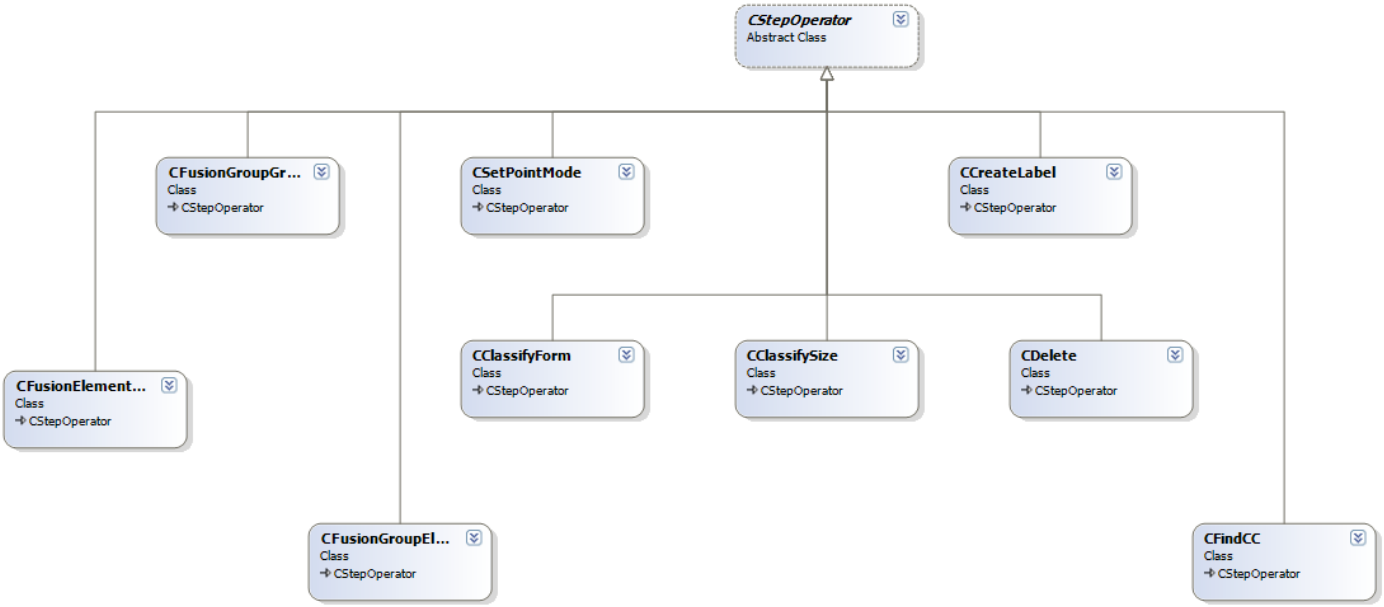
Goal = recognize an EOC regarding its shape.

- Operands : *EOC Parent, EOC Child, EOC New, Score*
- Function : insert *New* id *Score* is satisfy
- Algorithm
  1. Insert(Pattern with enableScaling)

CreateLabel

Goal = create a new label, that will be assigned to some futures EOC.

Level 1 operator class diagram



Level 2 Operators

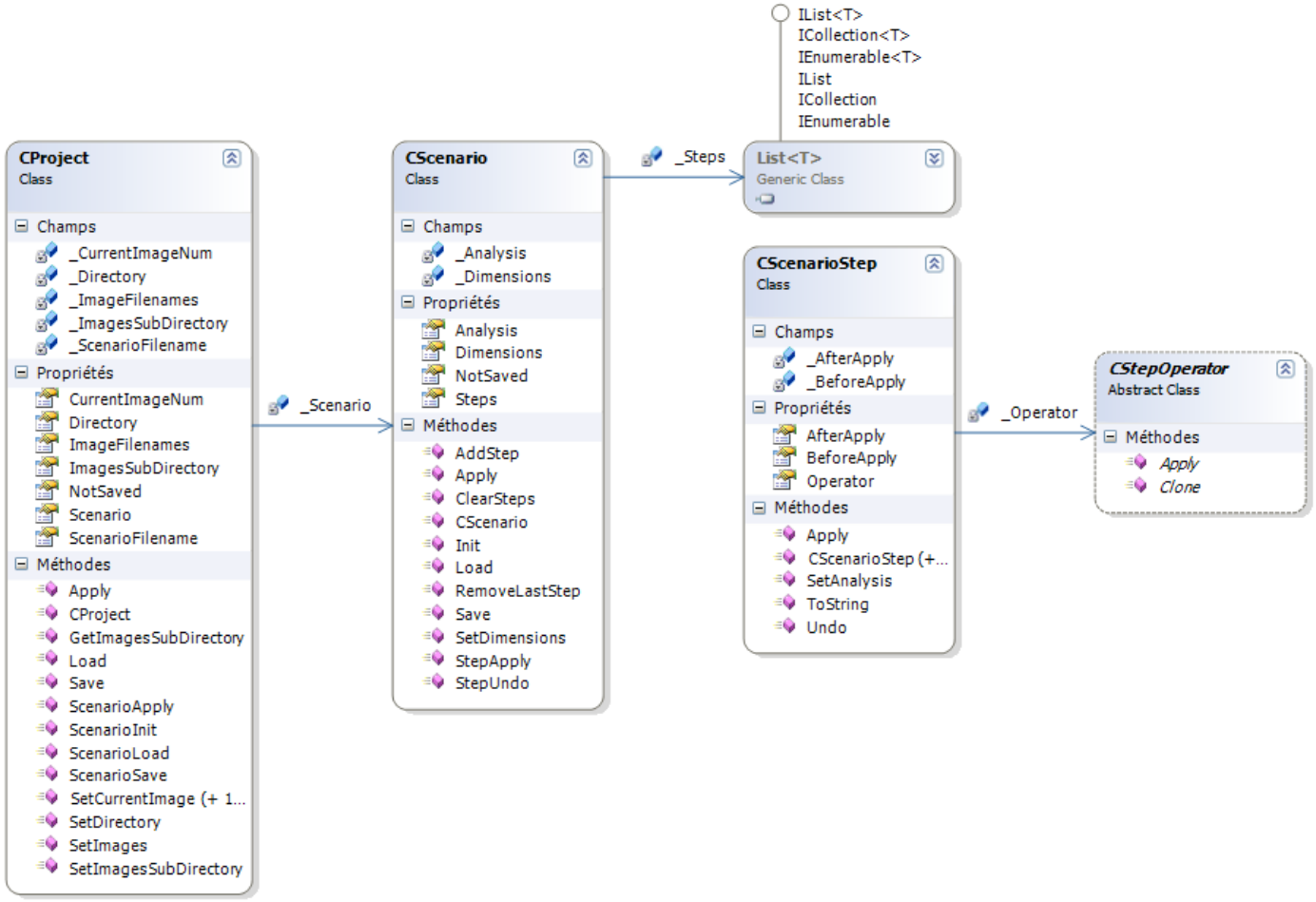
A level 2 operator is an algorithm that uses both level 1 and level 0 operators.

Only one exists for the time being.

DeleteIncludedEOC

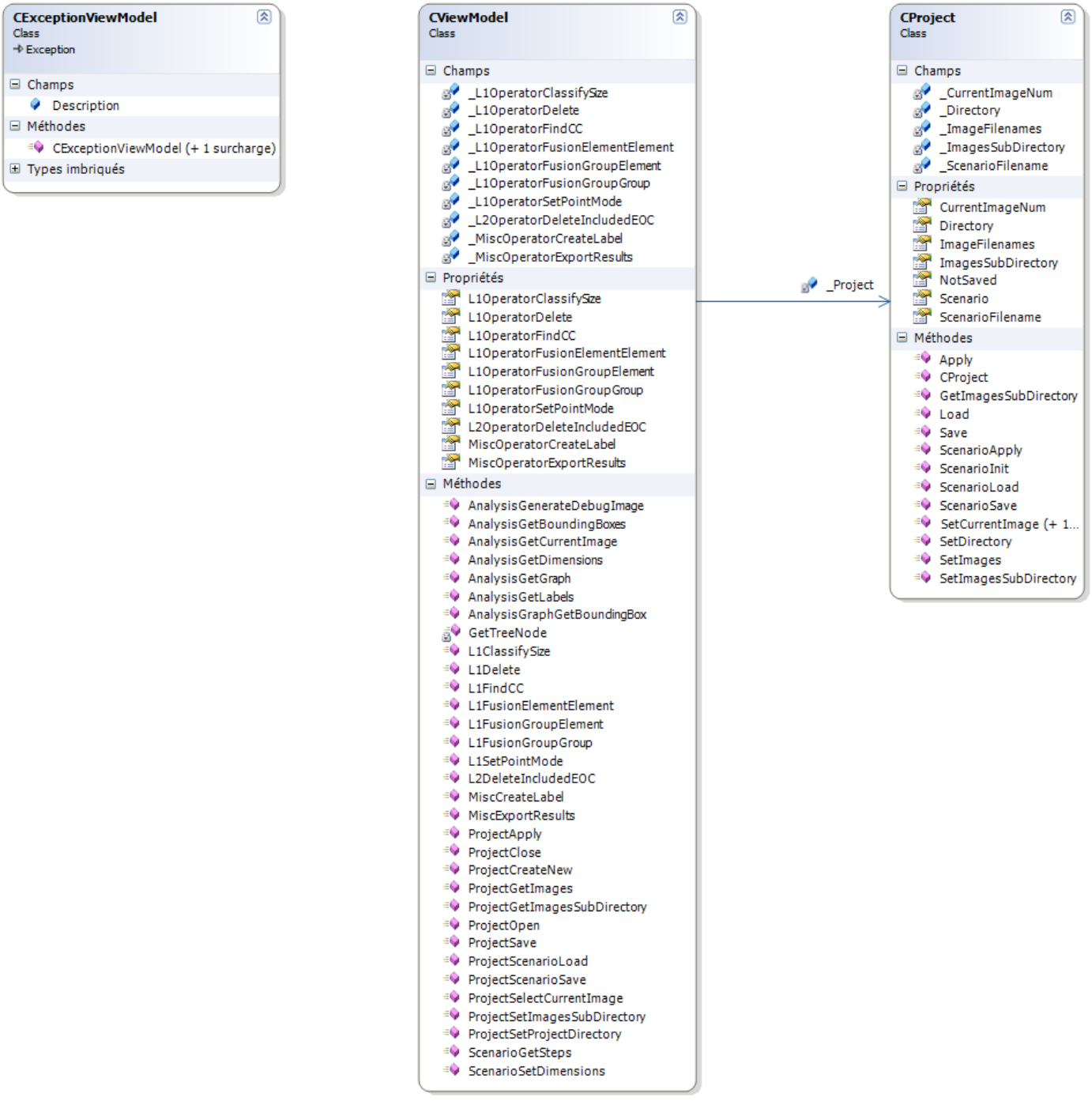
Goal = delete the EOC contained in another EOC

Scenario – Project



ViewModel

ViewModel allow to command the core of AGORA 3 and to display the results in a friendly GUI interface. Both display with windows and command line unit tests are available. ViewModel also implement an exception manager to validate the domain of parameters that is sent to the core of AGORA.



GUI

Implementation of several menus and dialog boxes that allow to display or get information, in order to send them to an instance of CViewModel.