



J-XMCDA

Présenté par :

Kakama Sall

Badreddine El Jemli

Elise Brochen

Sami Meskaoui

Sandra TUZA MANGARA

Noufissa Bencherif-Ouedghiri



SOMMAIRE

1/Introduction au projet

2/Présentation de Diviz

3/ Présentation de quelques fonctionnalités

4/Conclusion

Présentation du projet XMCDa

Objectif du projet

- Permettre l'invocation de services web XMCDa
- Conserver la compatibilité entre services et leur donner une spécificité

Les points clés du projet sont les fichiers au format XMCDa et les logiciels DIVIZ, git, maven, glassfish



XMCDa V1



XMCDa V2

Organisation du Groupe



Prénom:	Samy	Sandra	Elise	Kakama	Badredine	Noufissa
Fonctionnalités:						
Diviz		✓		✓		
Alts	✓		✓			
Crits					✓	✓
ServiceDescr	✓			✓		
Basic Servlets		✓			✓	
BasicClient			✓			✓
ObjectsXML	✓			✓		
GenSchema	✓					✓
Online				✓		✓
UseDB			✓		✓	
SetDB1			✓		✓	
Rédaction du rapport		✓				

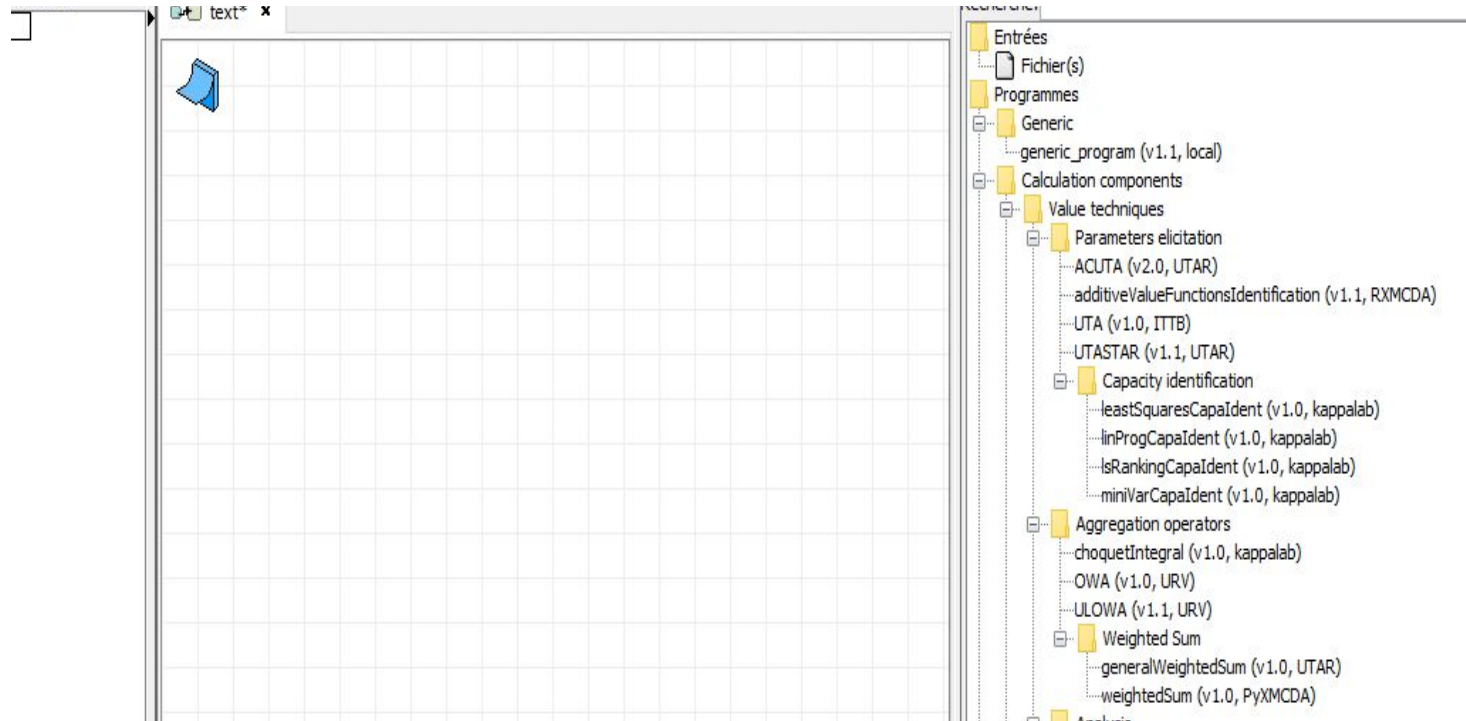
Présentation de DIVIZ



Quelques définitions:

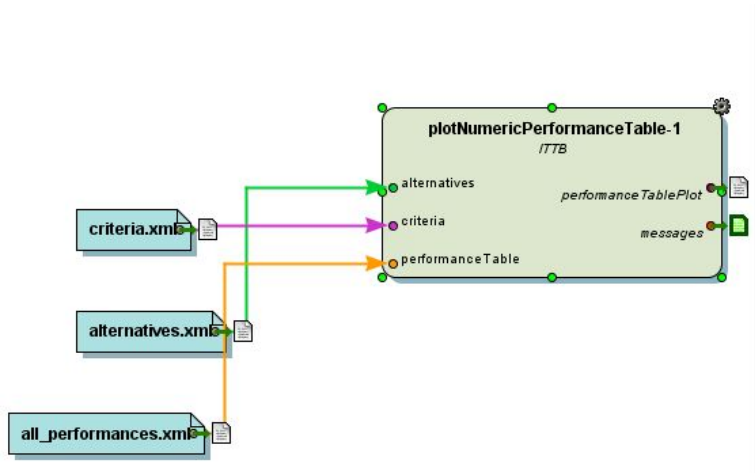
- ★ MCDA
- ★ DIVIZ

Présentation de la plateforme DIVIZ



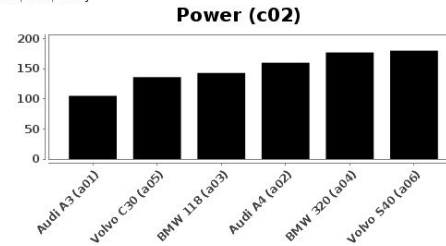
Présentation de DIVIZ

Exemple d'utilisation de DIVIZ



performance table plot

{ c02, c03, c04, c01, c05 }



J-XMCDA

Objectif

Permettre l'invocation de services web XMCDAs (spécialisés en aide multicritère à la décision) grâce à une interface graphique, fournir des outils divers pour faciliter leur usage. La connaissance de l'aide multicritère à la décision n'est pas requise.

Contexte

Une série de services web ouverts permet d'utiliser les méthodes d'aide multicritère à la décision. Ils reposent sur le format XMCDAs et sur le logiciel Diviz, qui permet l'invocation des services sans devoir programmer. L'utilisateur doit cependant fabriquer à la main les fichiers XML d'entrée (ou convertir du CSV), et plus généralement il n'existe pas d'intégration des outils dans un logiciel très simple d'emploi.

Le format d'encodage actuellement utilisé pour XMCDAs est XMCDAv2 (ci-après : X2). Tous les services web XMCDAs acceptent en entrée du X2 et sortent du X2. L'inconvénient de cette approche est que les services web sont mal décrits par le schéma. À l'opposé, le schéma xmcdas-modular a été proposé, qui ne doit pas être utilisé pour décrire un service web directement, mais bien pour générer des schémas qui servent à décrire des services web. Ainsi, on pourrait conserver la compatibilité entre services, mais leur donner une spécificité. Votre mission est (entre autres) d'assurer la compatibilité de ces deux approches, et l'invocation automatique de services webs dans la mesure du possible.

Fonctions demandées

Diviz

Un binôme se familiarise avec l'usage de Diviz, et le présente aux autres. Il faut suivre un tutoriel avec un cas d'usage non trivial ; télécharger les données six real cars et les utiliser sur un problème. Écrire un rapport avec ce qui a été et les problèmes rencontrés. (1)

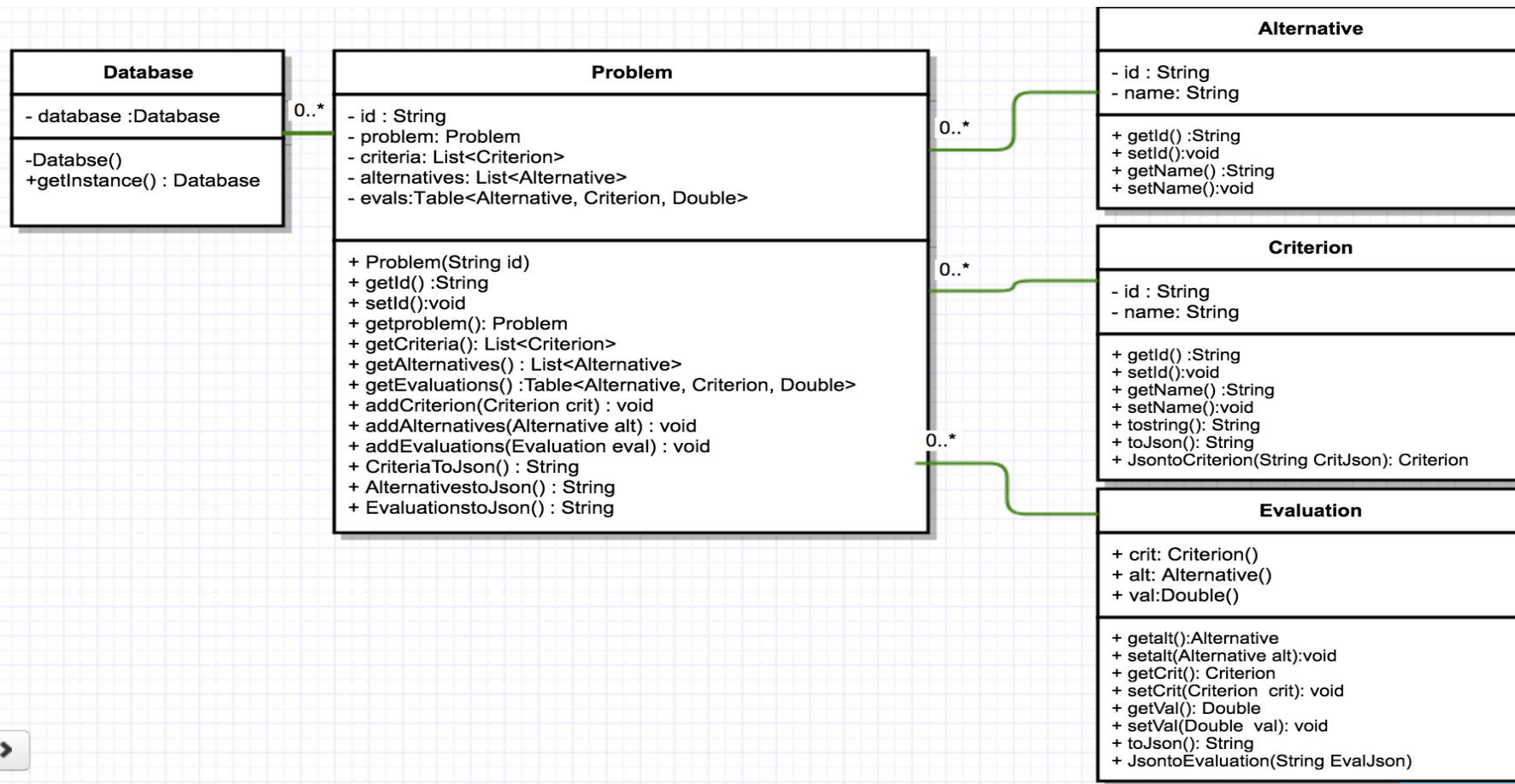
Alts

Objet Alternative (id: int, name: String) ; un objet pour représenter un rangement d'alternatives avec ex-æquo (association de chaque alternative à son rang). Conseil : les représenter par un DAG où la relation parent enfant est la relation « est moins bonne que » (suit dans le classement). Voir bibliothèque Guava. Ou par une liste de liste (liste d'alternatives ex-æquo). Encodage et décodage de Alternative en JSON, et utiliser le format de [{PrefLib}](#) pour la représentation textuelle d'un rangement avec ex-æquo (encodage et décodage). (1)

Crits

Objet Criterion (id et name) ; un objet Evaluations pour représenter les évaluations d'un ensemble d'alternatives, chacune sur une série de critère. Voir exemple six real cars. Conseil : utiliser une Table de Guava. Représentation JSON d'un critère, et d'un objet évaluations en CSV et JSON. (1)

UML des classes fondamentales XMCD



Criterion Class

The screenshot shows the Eclipse IDE with the following components:

- Package Explorer:** Shows the project structure with the package `io.github.oliviercailloux.y2017` and the class `CriterionTest`.
- JUnit Test Results:** Shows the test `io.github.oliviercailloux.y2017.CriterionTest` finished after 0.491 seconds with 1/1 runs, 0 errors, and 0 failures.
- Code Editor:** Displays the source code of `CriterionTest.java`. The code is as follows:

```
1 package io.github.oliviercailloux.y2017;
2
3 import org.junit.Test;
4
5 public class CriterionTest {
6
7     @Test
8     public void testToJson() {
9
10         Criterion criterion = new Criterion(1, "a");
11         String critJson = criterion.toJson();
12         System.out.println(critJson);
13
14     }
15 }
16
17 }
18
```
- Task List:** Shows a list of tasks, including "Thursday - Today", "Friday", "Saturday", "This Week", "Next Sunday", "Next Monday", "Next Tuesday", "Next Wednesday", and "Next Thursday".
- Outline:** Shows the class `io.github.oliviercailloux.y2017` and the method `testToJson() : void`.
- Console:** Shows the output of the test, which is a JSON object:

```
{
  "id": 1,
  "name": "a"
}
```
- Updates Available:** A small notification in the bottom right corner.

Evaluation Class :

The screenshot displays the Eclipse IDE interface with the following components:

- Package Explorer:** Shows the project structure with the package `io.github.oliviercailloux.y2017`. The test `io.github.oliviercailloux.y2017.EvaluationTest` is listed with a status of "Runs: 1/1", "Errors: 0", and "Failures: 0".
- JUnit View:** Displays the execution results of the test, indicating it finished after 0.402 seconds.
- Code Editor:** Shows the source code of `EvaluationTest.java`. The code defines a `public class EvaluationTest` with a `@Test` annotated `test()` method. The method creates a `Criterion` object, an `Alternative` object, an `Evaluation` object, and prints the evaluation result as JSON.
- Task List:** Displays a list of tasks, including "Thursday - Today", "Friday", "Saturday", "This Week", "Next Sunday", "Next Monday", "Next Tuesday", "Next Wednesday", and "Next Thursday".
- Outline:** Shows the class hierarchy, including `io.github.oliviercailloux.y2017` and `EvaluationTest`, with the `test() : void` method listed.
- Console:** Displays the output of the test execution, showing the JSON representation of the evaluation result:

```
<terminated> EvaluationTest [JUnit] /Library/Java/JavaVirtualMachines/jdk1.8.0_73.jdk/Contents/Home/bin/java (Feb 8, 2018, 9:32:50 PM)
{
  "crit": {
    "id": 1,
    "name": "c"
  },
  "val": 1.0
}
```
- Updates Available:** A notification in the bottom right corner stating "Updates are available for your software. Click to review and install updates."

Problem Class:

The screenshot displays the Eclipse IDE interface with the following components:

- Package Explorer (Left):** Shows the project structure. The 'src/test/java' directory is expanded, showing the 'io.github.oliviercailloux.y2017' package. The 'ProblemTest.java' file is selected.
- Editor (Center):** Displays the code for 'ProblemTest.java'. The code is as follows:

```
1 package io.github.oliviercailloux.y2017;
2
3
4
5 import org.junit.Test;
6
7 public class ProblemTest {
8
9     @Test
10    public void test() {
11        Problem pb = new Problem("01");
12        Criterion crit0= new Criterion(1,"c");
13        Alternative alt0 = new Alternative("1","a");
14        Criterion crit1= new Criterion(2,"c");
15        Alternative alt1 = new Alternative("2","a");
16        Evaluation eval0 = new Evaluation(alt0, crit0, 1.0);
17        Evaluation eval1= new Evaluation(alt0, crit1, 2.0);
18        Evaluation eval2 = new Evaluation(alt1, crit0, 3.0);
19        Evaluation eval3 = new Evaluation(alt1, crit1, 4.0);
20        pb.addCriterion(crit0);
21        pb.addCriterion(crit1);
22        pb.addAlternatives(alt0);
23        pb.addAlternatives(alt1);
24        pb.addEvaluations(eval0);
25        pb.addEvaluations(eval1);
26        pb.addEvaluations(eval2);
27        pb.addEvaluations(eval3);
28        System.out.println(pb.AlternativestoJson());
29        System.out.println(pb.CriteriatoJson());
30        System.out.println(pb.EvaluationstoJson());
31    }
32 }
```
- Task List (Top Right):** Shows a calendar view for Thursday - Today, Friday, Saturday, and This Week.
- Outline (Bottom Right):** Shows the project structure, with 'ProblemTest' selected under 'io.github.oliviercailloux.y2017'. The method 'test() : void' is listed.
- Console (Bottom):** Shows the output of the test execution. The output is as follows:

```
<terminated: EvaluationTest [JUnit] /Library/Java/JavaVirtualMachines/jdk1.8.0_73.jdk/Contents/Home/bin/java (Feb 8, 2018, 9:32:50 PM)
{,
  "crit": {
    "id": 1,
    "name": "c"
  },
  "val": 1.0
}
```
- Updates Available (Bottom Right):** A small window indicating that updates are available for the software.

AlternativeRankingTest Class:

The screenshot shows an IDE with the `AlternativeRankingTest.java` file open. The code defines a test class that creates four `Alternative` objects, uses an `AlternativeRanking` instance to rank them, and prints the resulting JSON map. The console output shows the execution of the test, including the printed JSON data.

```
1 package io.github.oliviercailloux.y2017;
2 import org.junit.Test;
3
4
5 public class AlternativeRankingTest {
6
7     @Test
8     public void test() {
9         Alternative a1=new Alternative("a1","volvo");
10        Alternative a2=new Alternative("a1","volvo");
11        Alternative a3=new Alternative("a1","volvo");
12        Alternative a4=new Alternative("a1","volvo");
13
14        AlternativeRanking alts = new AlternativeRanking();
15        try {
16            alts.altsRanking(1, a1);
17            alts.altsRanking(1, a4);
18            alts.altsRanking(2, a3);
19        } catch (AlternativeException e) {
20            // TODO Auto-generated catch block
21            e.printStackTrace();
22        }
23
24        alts.displayMap(alts.getMulti_ranking());
25        System.out.println(alts.encodeJson(alts.getMulti_ranking()));
26    }
27
28
29
30 }
```

Console Output:

```
<terminated> AlternativeRankingTest (1) [JUnit] /usr/lib/jvm/java-8-openjdk-i386/bin/java (13 févr. 2018 11:19:41)
1:volvo
1:volvo
2:volvo
{"alternatives":[{"rank":1,"name":"volvo","id":"a1"}, {"rank":1,"name":"volvo","id":"a1"}, {"rank":2,"name":"volvo","id":"a1"}]}
```

AlternativeRankingTest Class:

The screenshot shows an IDE window with the `AlternativeRankingTest.java` file open. The code defines a test method `test()` that creates four `Alternative` objects, initializes an `AlternativeRanking` instance, and performs several ranking operations. The IDE's right sidebar shows the `Outline` view with the `test() : void` method selected. The bottom status bar indicates that the test was executed successfully with 0 errors and 0 failures, taking 0.059 seconds.

```
ve.java  AlternativeRanking.java  AlternativeRankingTest.java  X
je io.github.oliviercailloux.y2017;
t org.junit.Test;

: class AlternativeRankingTest {
    @Test
    public void test() {
        Alternative a1=new Alternative("a1","volvo");
        Alternative a2=new Alternative("a1","volvo");
        Alternative a3=new Alternative("a1","volvo");
        Alternative a4=new Alternative("a1","volvo");

        AlternativeRanking alts = new AlternativeRanking();
        try {
            alts.altsRanking(1, a1);
            alts.altsRanking(1, a4);
            alts.altsRanking(2, a3);
        } catch (AlternativeException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        alts.displayMap(alts.getMulti_ranking());
        System.out.println(alts.encodeJson(alts.getMulti_ranking()));
    }
}
```

Outline

- io.github.oliviercailloux.y2017
- AlternativeRankingTest
 - test() : void

er 0,087 seconds

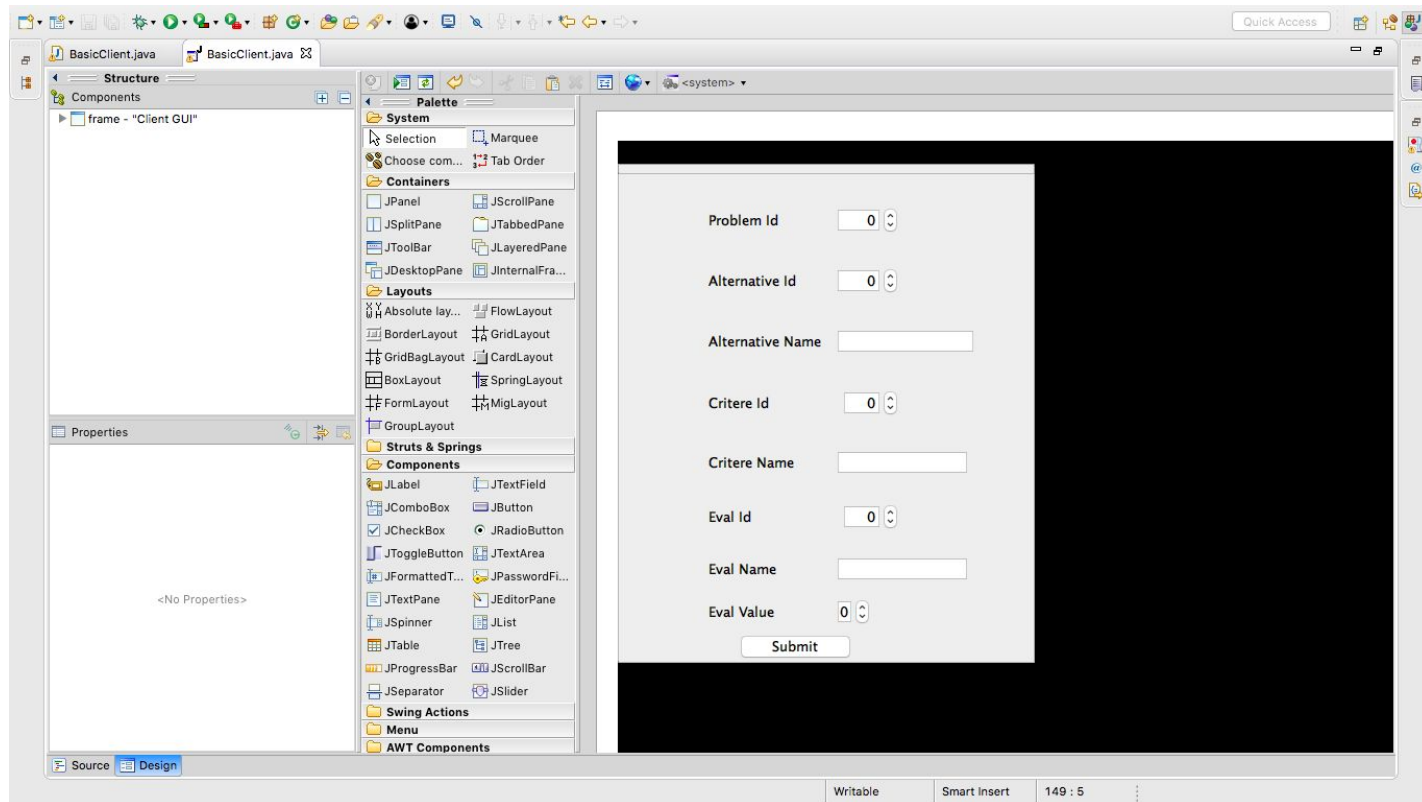
Errors: 0 Failures: 0

b.oliviercailloux.y2017.AlternativeRankingTest [Runner: JUnit 4] (0,059 s)

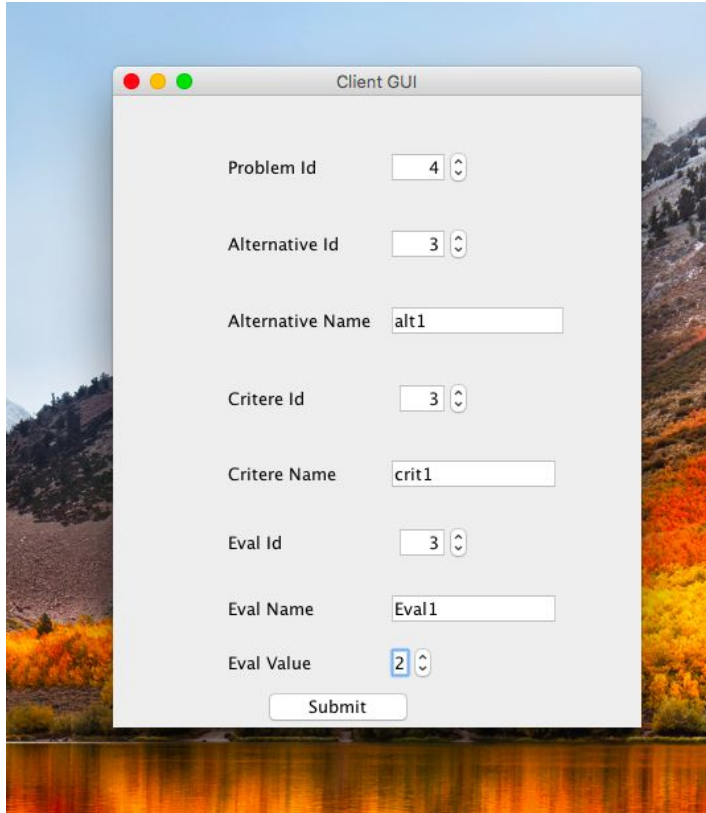
Failure Trace

Interface Utilisateur : Basic Client

- ★ GUI
- ★ SWING



Interface Utilisateur : Basic Client



The image shows a 'Client GUI' window with the following fields and values:

Field	Value
Problem Id	4
Alternative Id	3
Alternative Name	alt1
Critere Id	3
Critere Name	crit1
Eval Id	3
Eval Name	Eval1
Eval Value	2

At the bottom of the window is a 'Submit' button.

WebServiceDesc Class

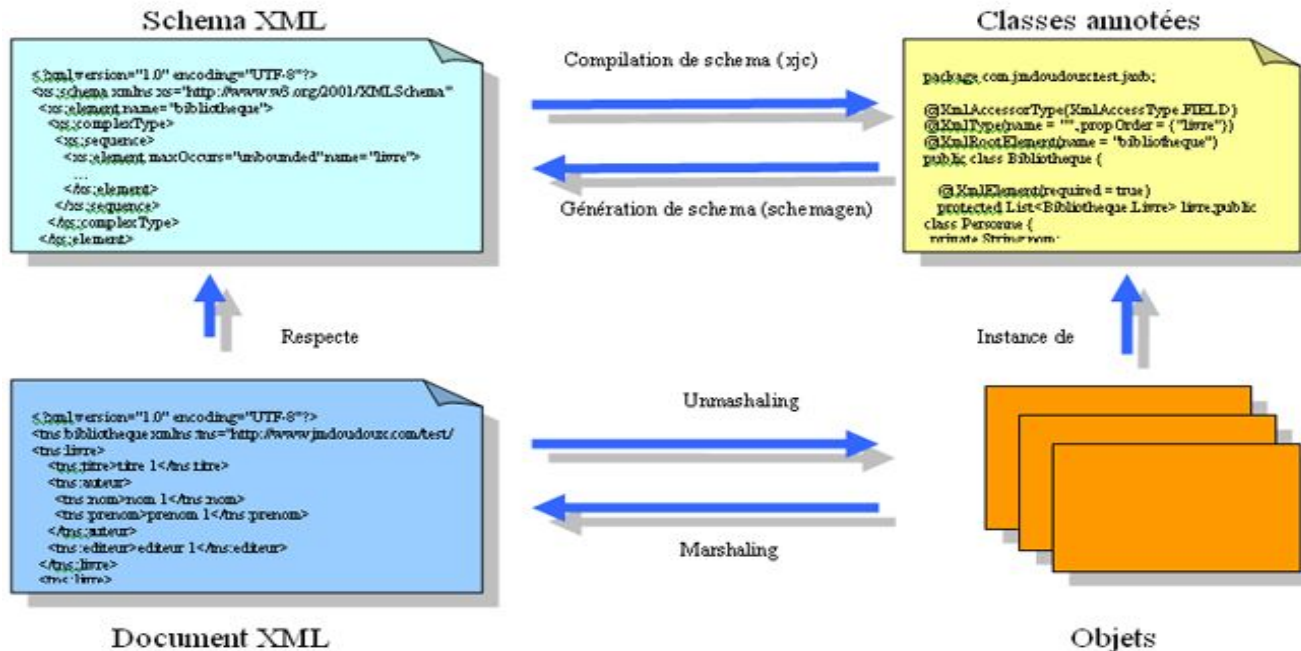
```
public enum XmcdModularTypes {  
  
    DEFAULTX2, FUNCTION, EXPONENTIALFUNCTION, POINT2D, PIECEWISELINEARFUNCTION, AFFINELINEARFUNCTION, KEYEDENTITY, ALTERNATIVE,  
    ALTERNATIVESSET, CATEGORY, CATEGORYSET, ATTRIBUTE, ATTRIBUTESET, NOMINALTOCARDINAL, NOMINALTOCARDINALSET, MEASUREMENT, EXACTME  
    BINARYMEASUREMENT, INTERVAL, GAUSSIAN, IMPRECISENOMINALMEASUREMENT, CRITERION, CRITERIONSET, DIRECTUTILITYCRITERION,  
    DIRECTUTILITYCRITERIONSET, NOMINALUTILITYCRITERION, NOMINALUTILITYCRITERIONSET, CARDINALUTILITYCRITERION, CARDINALUTILITYCRI  
    DIRECTEDCRITERION, DIRECTEDCRITERIONSET, OUTRANKINGCRITERIONSET, VALUEDENTITY, VALUEDENTITYSET, EXACTVALUEDENTITY, EXACTVALUE  
    ANYVALUEDPAIR, ANYVALUEDRELATION, BINARYVALUEDPAIR, BINARYRELATION, VALUEDPAIR, VALUEDRELATION, INTERVALVALUEDPAIR, INTERVALV  
    GAUSSIANVALUEDPAIR, GAUSSIANVALUEDRELATION, NOMINALVALUEDPAIR, NOMINALVALUEDRELATION, IMPRECISENOMINALVALUEDPAIR, IMPRECISENG  
  
}  
  
8 public class WebServiceDescrTest {  
9     @Test  
10     public void test() {  
11         Map<String, XmcdModularTypes> map= new HashMap<>();  
12         WebServiceDescription w=new WebServiceDescription(map);  
13         map.put("id1", XmcdModularTypes.ATTRIBUTE);  
14         map.put("id2", XmcdModularTypes.CRITERION);  
15  
16         w.setMap(map);  
17         w.displayMap(map);  
18         System.out.println(w.encodeJson(map));  
19     }  
20 }  
21
```

Console Markers Properties Servers Data Source Explorer Snippets Problems JUnit

```
<terminated> WebServiceDescrTest (1) [JUnit] /usr/lib/jvm/java-8-openjdk-1386/bin/java (13 févr. 2018 12:35:07)  
id2 : CRITERION  
id1 : ATTRIBUTE  
{"id2":"CRITERION","id1":"ATTRIBUTE"}
```

Schema Generator and Object XML Classes

→ JAXB (Java Architecture for XML Binding)



Schema Generator and Object XML Classes

```
4
30 import java.util.ArrayList;
9
10 @XmlRootElement(name = "alternatives")
11 @XmlAccessorType(XmlAccessType.FIELD)
12 public class Alternatives {
13
14     @XmlElement(name = "alternative")
15     private ArrayList<Alternative> alternatives = null;
16
17     public ArrayList<Alternative> getAlternatives() {
18         return alternatives;
19     }
20
21     public void setAlternatives(ArrayList<Alternative> alternatives) {
22         this.alternatives = alternatives;
23     }
24
25
26
27 }
28
```

```
ObjectXmlTest.java  all_alternatives.xml  ObjectXml.java  SchemaGenerator.java
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <xm:alternatives xmlns:xm="http://xmcd-modular.github.io/2017/xsd">
3     <alternative>
4         <id>a1</id>
5         <name>volvo</name>
6     </alternative>
7     <alternative>
8         <id>a2</id>
9         <name>bmw</name>
10    </alternative>
11    <alternative>
12        <id>a3</id>
13        <name>mercedes</name>
14    </alternative>
15 </xm:alternatives>
```

Les Servlets



Jax-rs

@PATH, @GET, @PUT, @POST,
@Produces, @Consumes et @PathParam

- ★ **Déploiement des Web Services**

La base de données

- ★ MySQL+ l'interface Workbench
- ★ Driver MySql Connector
- ★ Hibernate



Conclusion

Problèmes rencontrés:

- Comprendre le sujet
- Manque de visibilité sur l'architecture du projet
- Génération de JSON lorsque l'objet est une Multimap
- Gestion des conflits lors d'un push sur Git
- Problème de mauvais dépôts sur la branche master de GIT