Présentation de notre projet :

SUBOO

Présenté par:
- NARTZ Kevin
- MAILLOS Sebastien
- MICHELETTI Quentin
- DJEGHRI Amine
- GHERIBI Lotfi
- RIDEY Guillaume

Team BLOUGE -

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Décembre 2019

Sommaire

Introduction

Diagramme de classe Composant

Diagramme de classe Composant

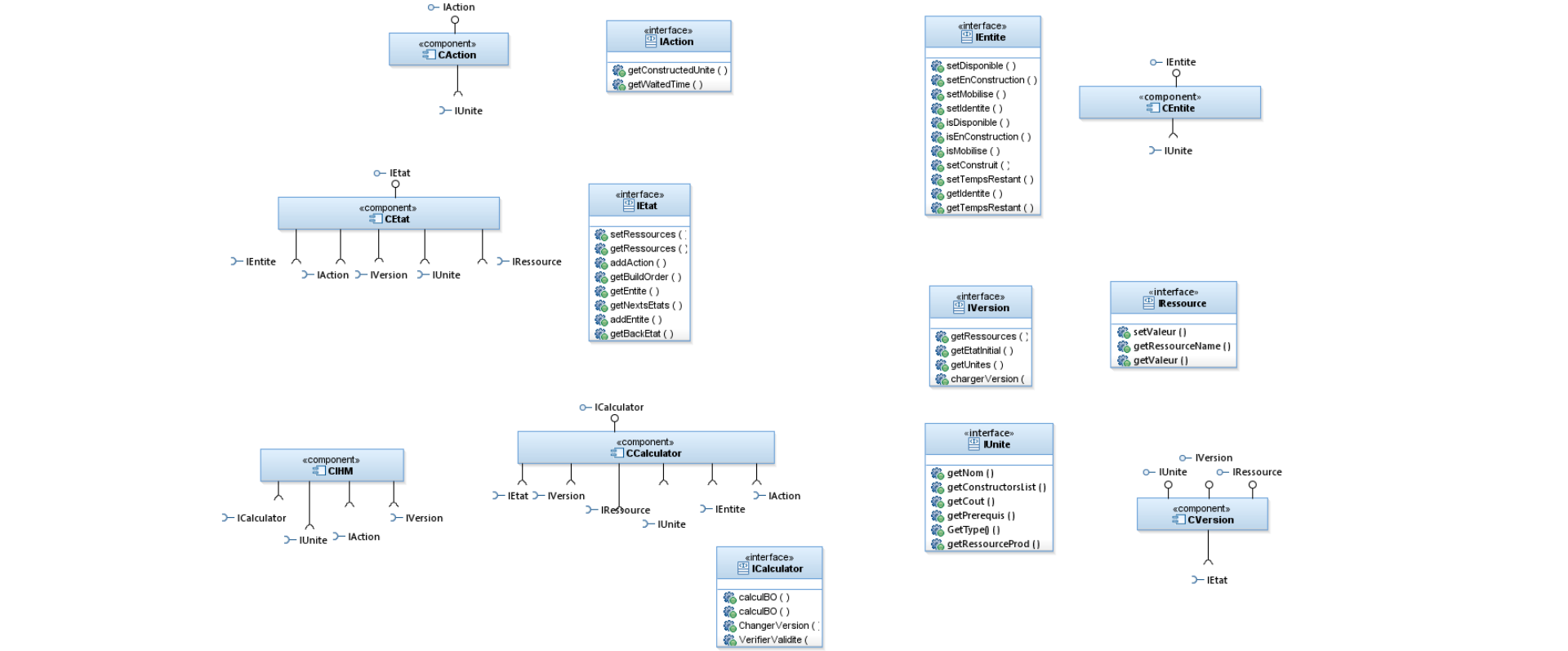
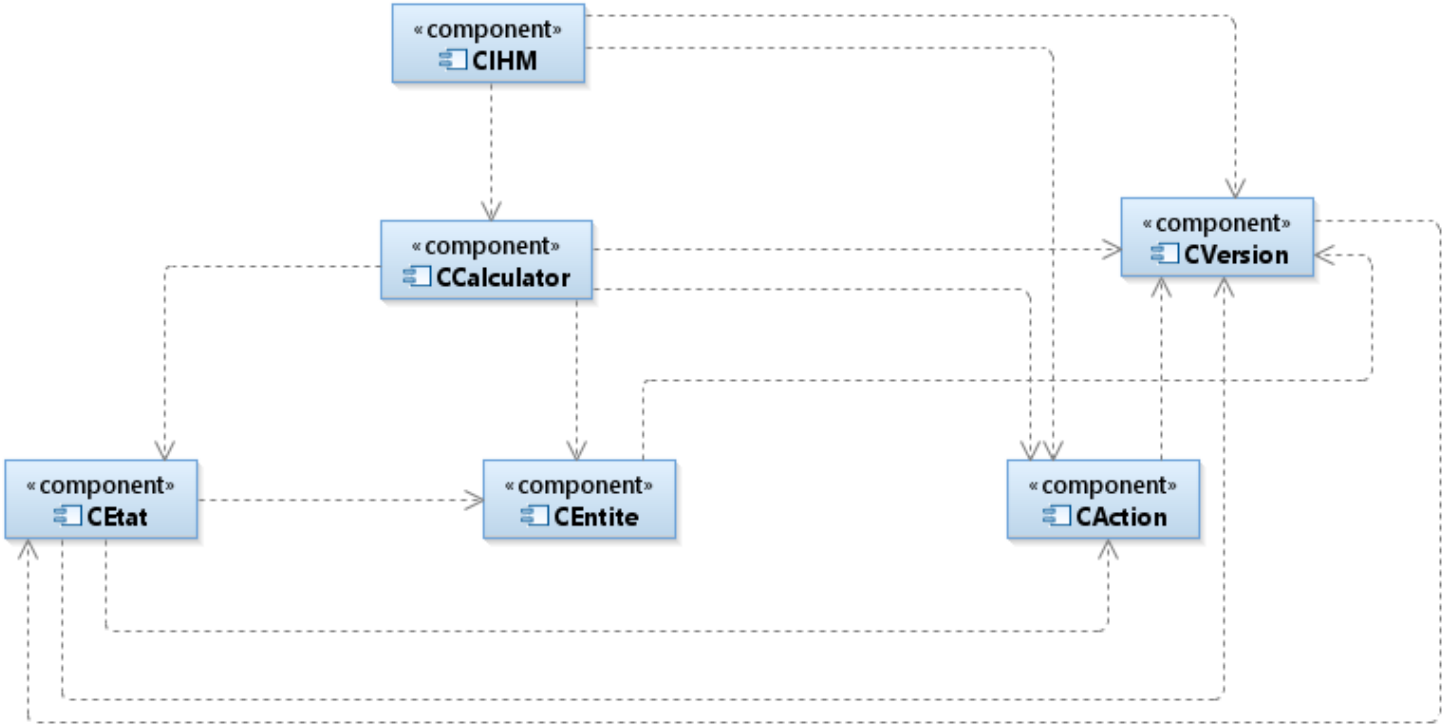


Diagramme d’instance nominale

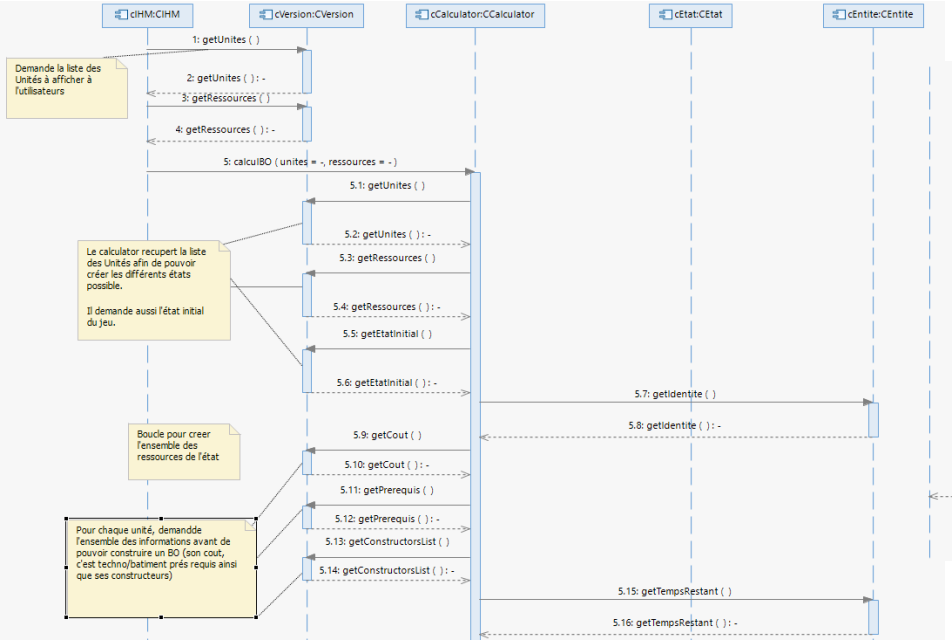
Diagramme d’instantiation nominale

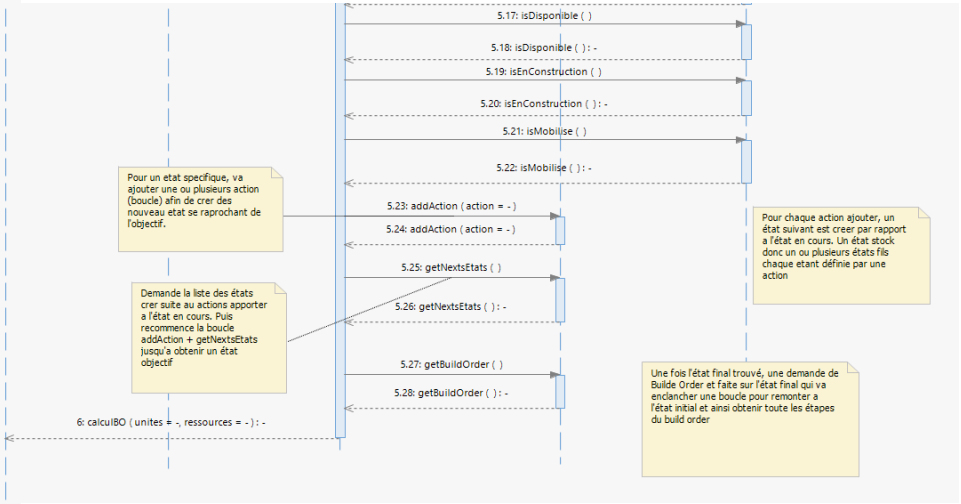


Rechercher BO:

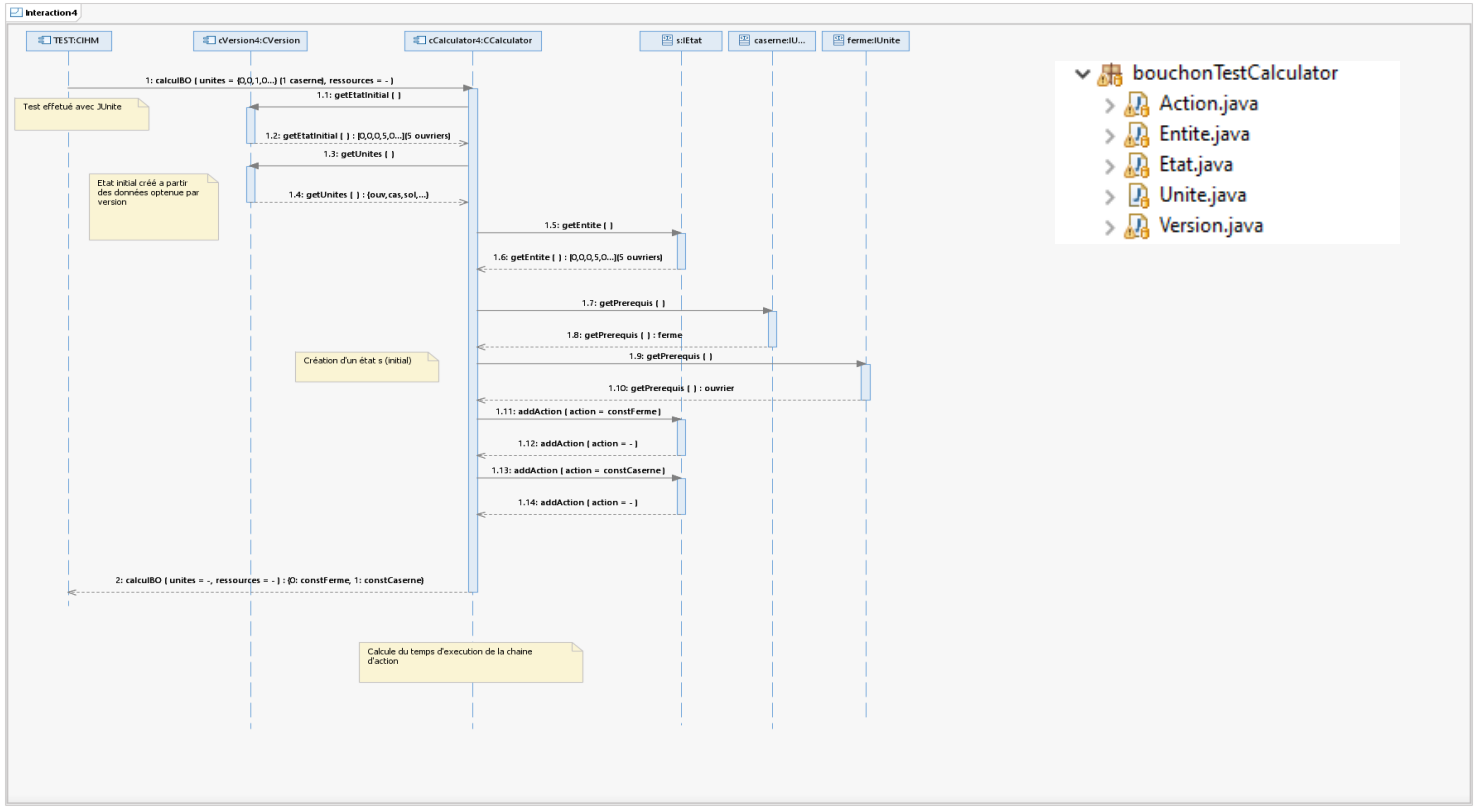
Diagramme de sequence inter-composant & tests d’integration

Rechercher BO:
Diagramme de séquence inter-composant





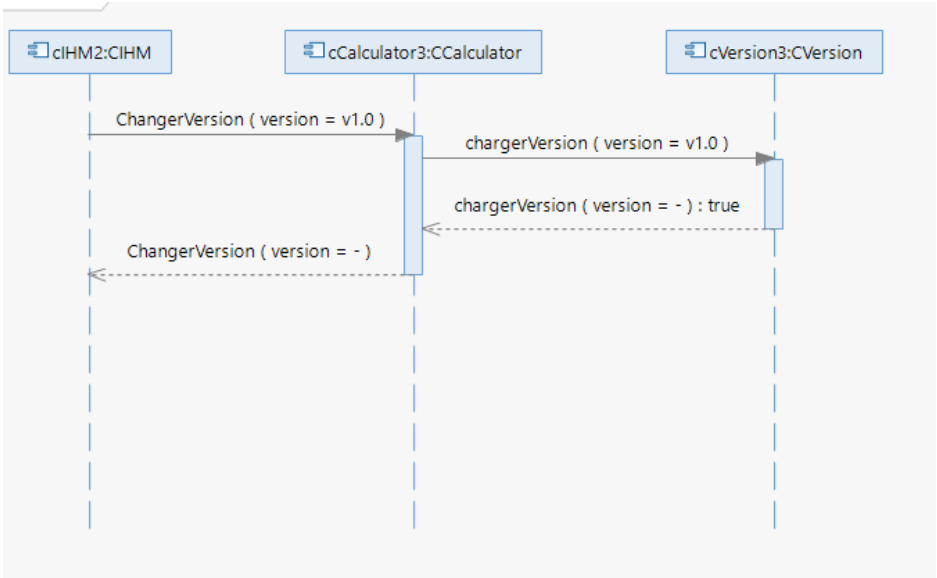
Rechercher BO:
Test d'intégration



Changer version:

Diagramme de sequence inter-composant & tests d’intégration

Changer version:
Diagramme de séquence inter-composant




```
Or,Nourriture
Centre,Ferme,Caserne,Hall,Ouvrier,Soldat,Boss
200;Batiment;Ouvrier;Ouvrier;Nourriture 6;Or 300
60;Batiment;;Ouvrier;Nourriture 6;Or 100
100;Batiment;Ouvrier Ferme;Ouvrier;;Or 150
120;Batiment;Ouvrier Caserne;Ouvrier;;Or 200
30;Unite;Centre;Centre;Or 1;Or 50,Nourriture 1
50;Unite;Caserne;Caserne;;Or 100, Nourriture 1
100;Unite;Hall Caserne;Caserne;;Or 200, Nourriture 3
9 2
50,0,1,0,0,0,5,0,0
```

Changer version:
Test d'intégration

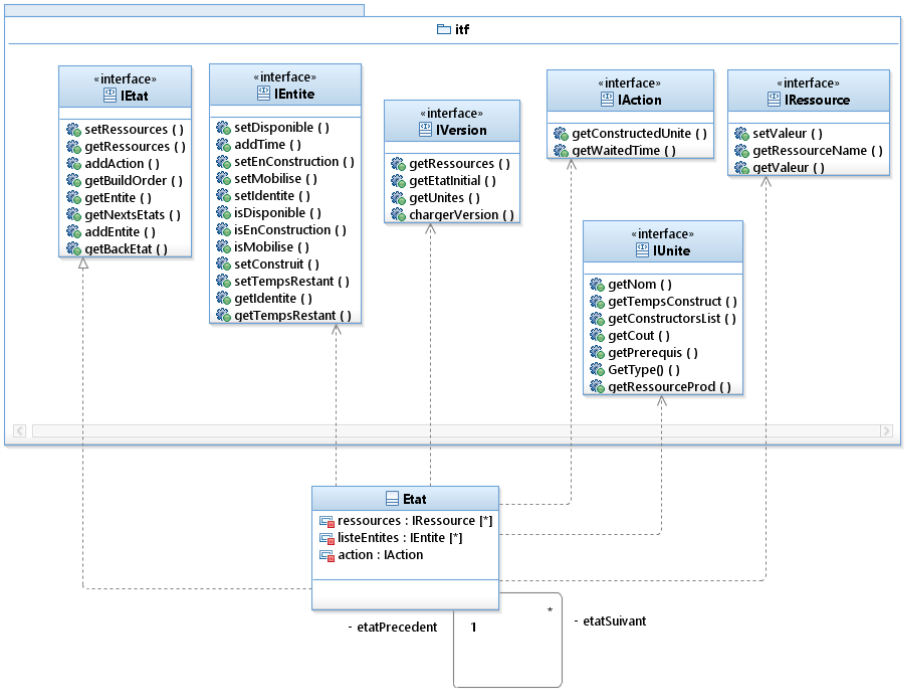
```java
    @Test
    public final void testFile() {
        IUnite      ferme   = new Unite("Ferme");
        IUnite      caserne = new Unite("Caserne");
        IUnite      centre  = new Unite("Centre");
        IUnite      hall    = new Unite("Hall");
        IUnite      ouvrier = new Unite("Ouvrier");
        IUnite      soldat  = new Unite("Soldat");
        IUnite      boss    = new Unite("Boss");
        IRessource ir = new Ressource("Nourriture", 0);
        IUnite  u = new Unite("Soldat");
        List<IUnite>     liu = new ArrayList<>();
        List<IRessource> lir = new ArrayList<>();
        lir.add(new Ressource("Or", 0));
        lir.add(new Ressource("Nourriture", 0));
        liu.add(centre);
        liu.add(ferme);
        liu.add(caserne);
        liu.add(hall);
        liu.add(ouvrier);
        liu.add(soldat);
        liu.add(boss);
        System.out.println(lir.get(1) + " " + ir);


        //On test les méthodes equals
        assertEquals(true, ir.equals(lir.get(1)));
        assertEquals(true, u.equals(soldat));
```

```java

        IVersion v;

        try {
            v = version.Version.getInstanceIVersion();

            // ON vérifie que la liste des unités et des ressources de base soient correctes
            assertEquals(true, equalList(lir, v.getRessources()));
            assertEquals(true, equalList(liu, v.getUnites()));
            lir.get(0).setValeur(50);
            // on vérifie que l'état initial est correcte
            assertEquals(lir, v.getEtatInitial().getRessources());
        } catch (IOException e) {
            e.printStackTrace();
        }
        }

    private <V> boolean equalList ( List<V> l, List<V> k) {
        int len = l.size();
        if(l.size() != k.size())
            return false;
        for(int i = 0; i < len ; i++) {
            if(!l.get(i).equals(k.get(i)))
                return false;
        }
        return true;
    }
}
```

Diagramme classe niveau conception détaillée

Diagramme classe niveau conception détaillée

Composant Etat:



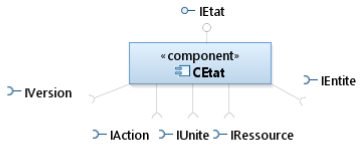


Diagramme classe niveau conception détaillée

Composant Version:

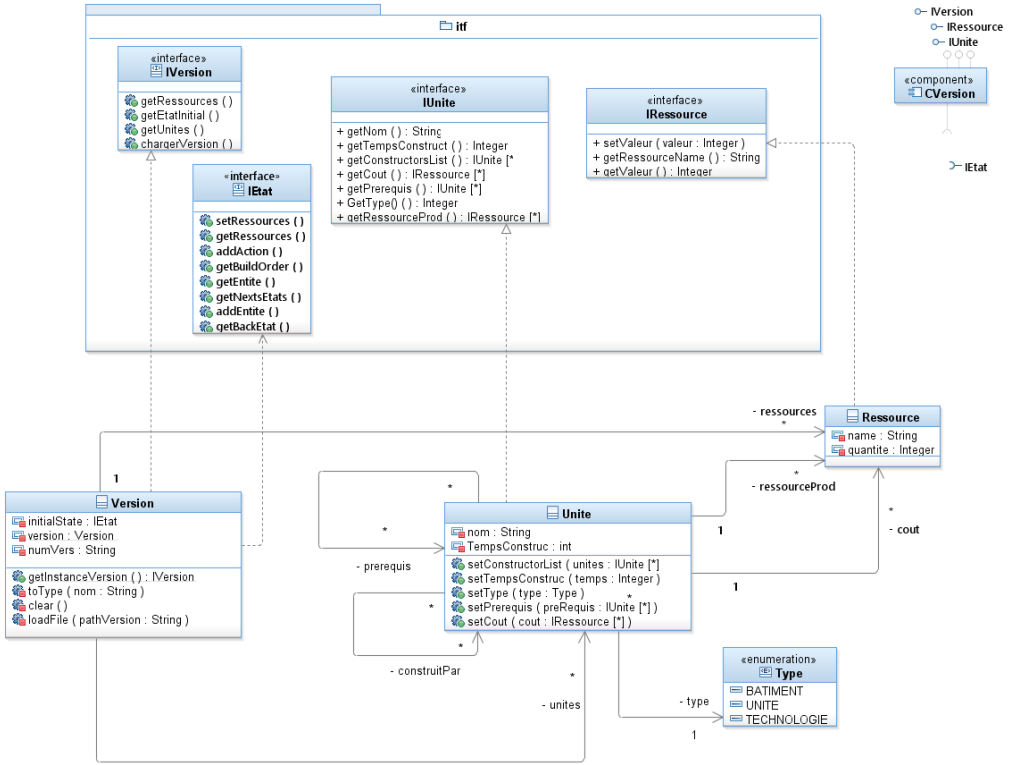


Diagramme classe niveau conception détaillée

Composant Entité:

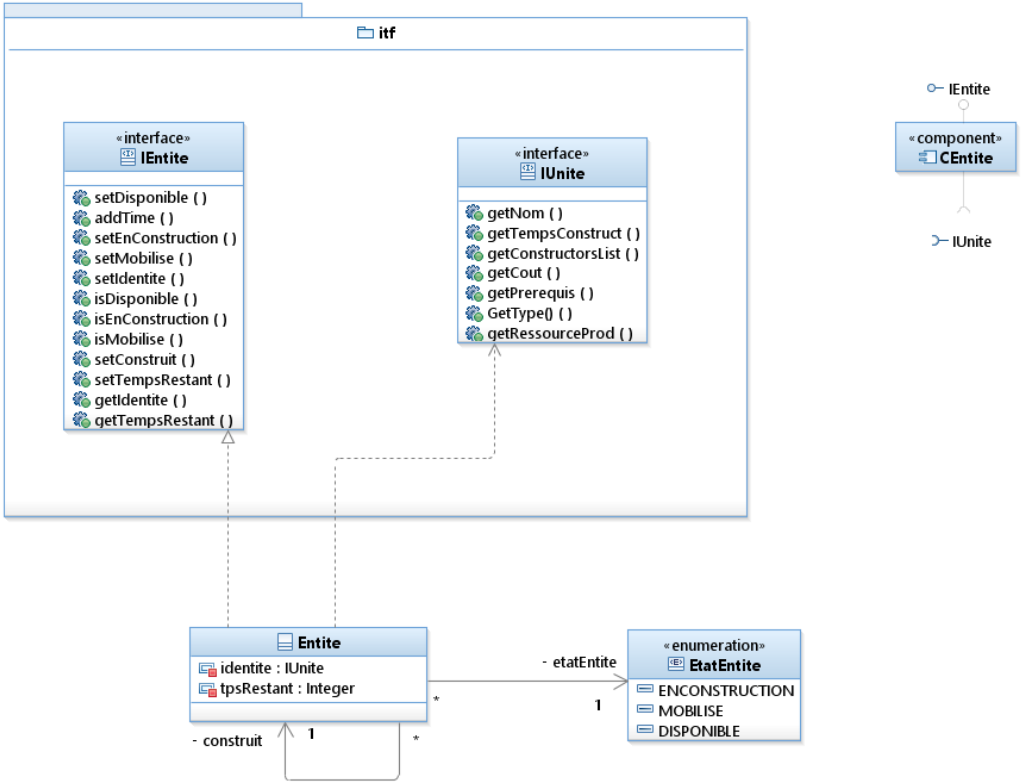
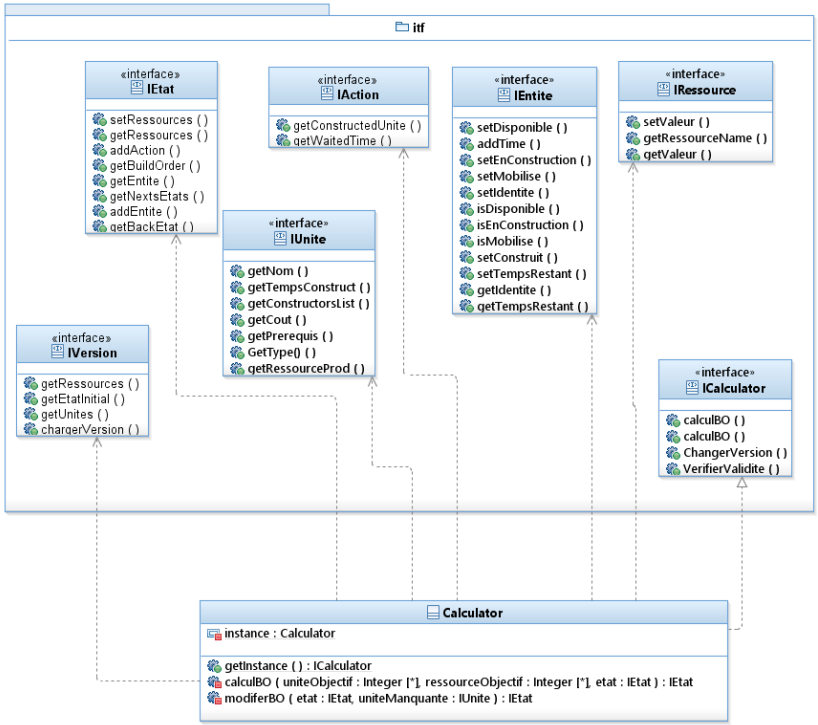


Diagramme classe niveau conception détaillée

Composant Calculator:



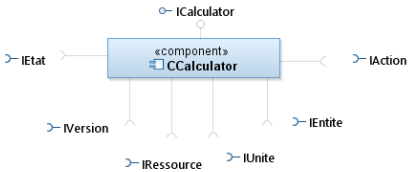


Diagramme classe niveau conception détaillée

Composant Action:

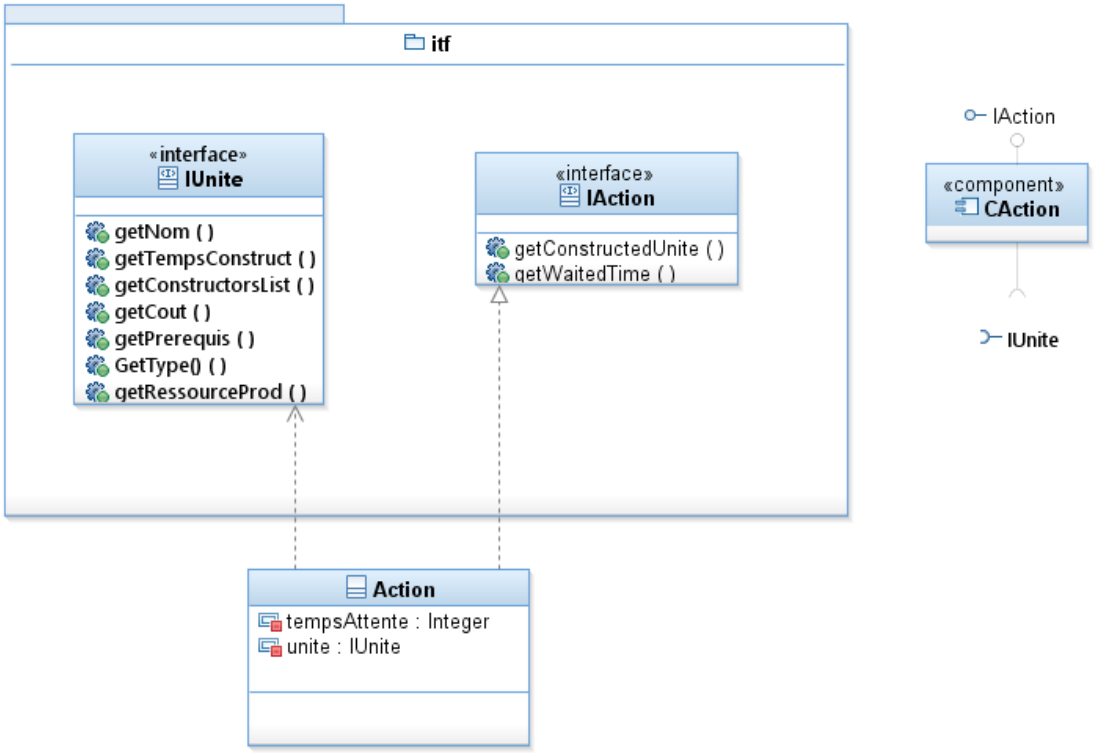
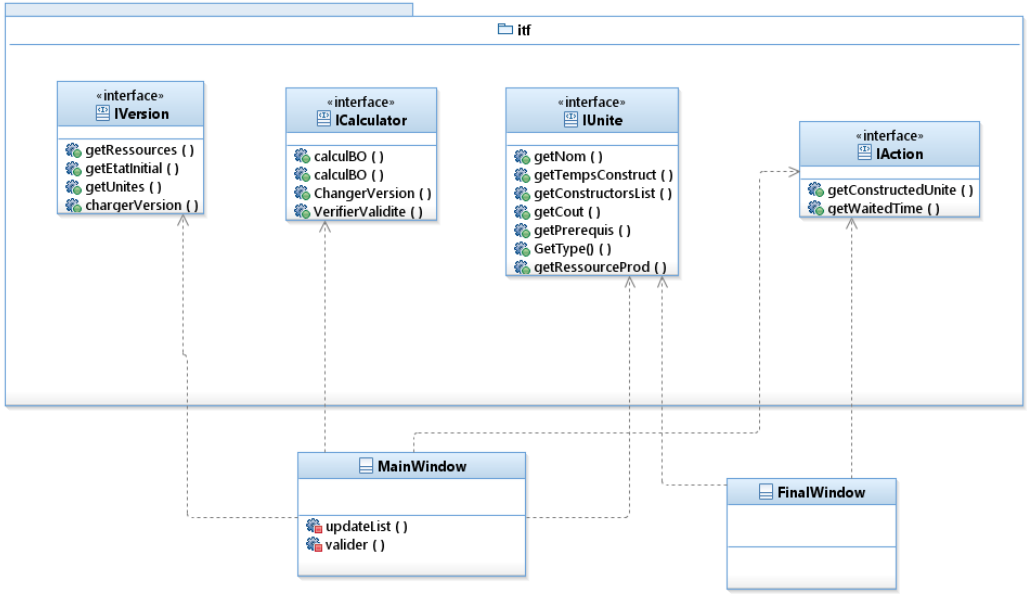
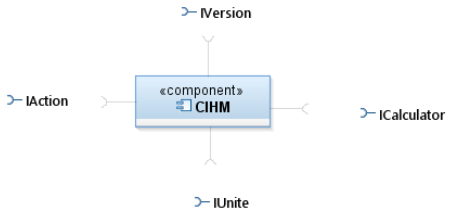


Diagramme classe niveau conception détaillée

Composant IHM:





Démonstration

Conclusion

Conclusion

- Difficultés :
  - Les passages code-modèle ,modèle-code
  - Implémentation des bouchons

- Bilan:
  - Découpe en composants
  - Répartition du travail


Merci
pour votre
attention